Einführung in Visual Computing

186.822

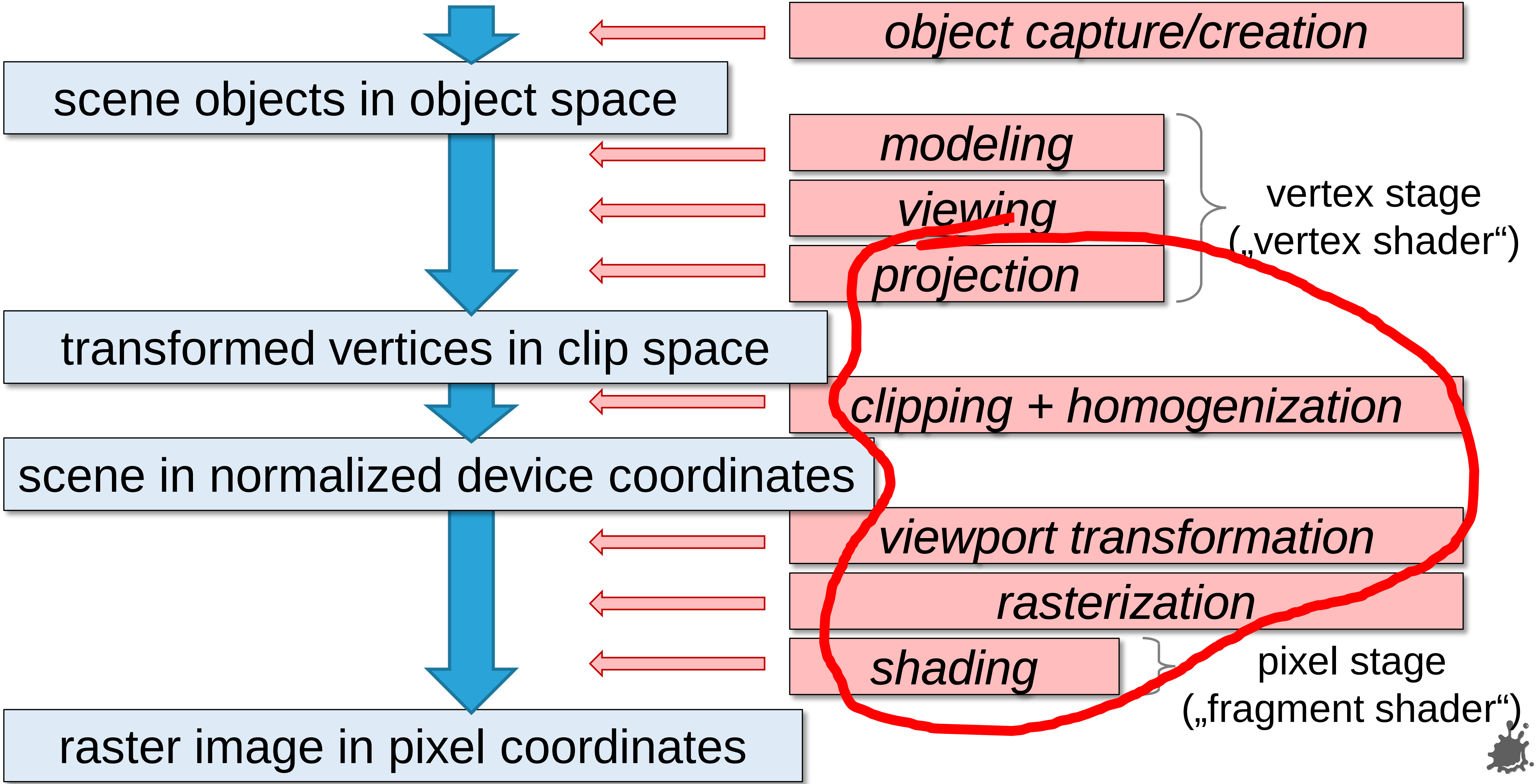
Ray Tracing

Werner Purgathofer

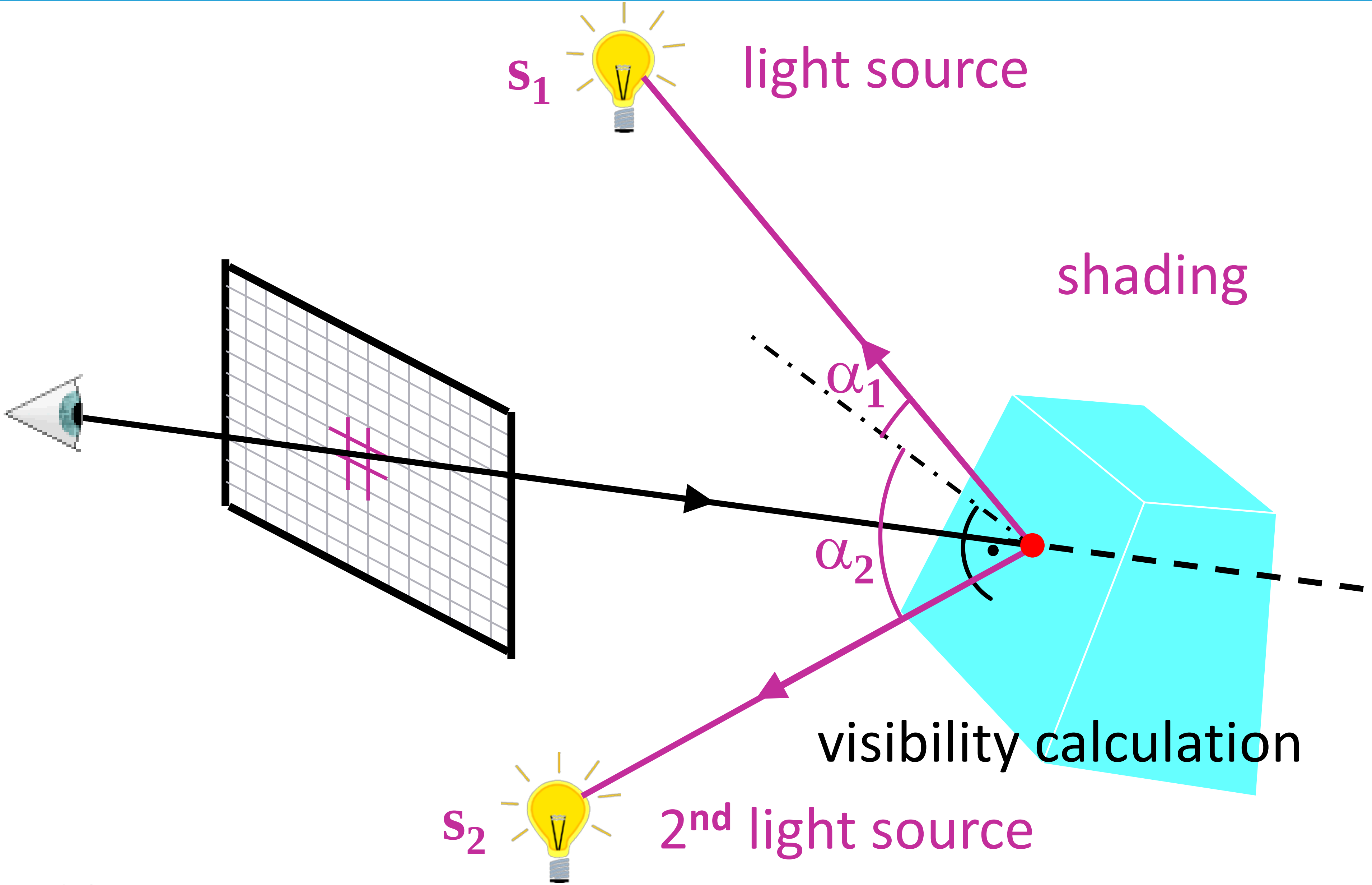
Surface-Rendering Methods

- polygon rendering methods
- **ray tracing**
- global illumination
- environment mapping
- texture mapping
- bump mapping

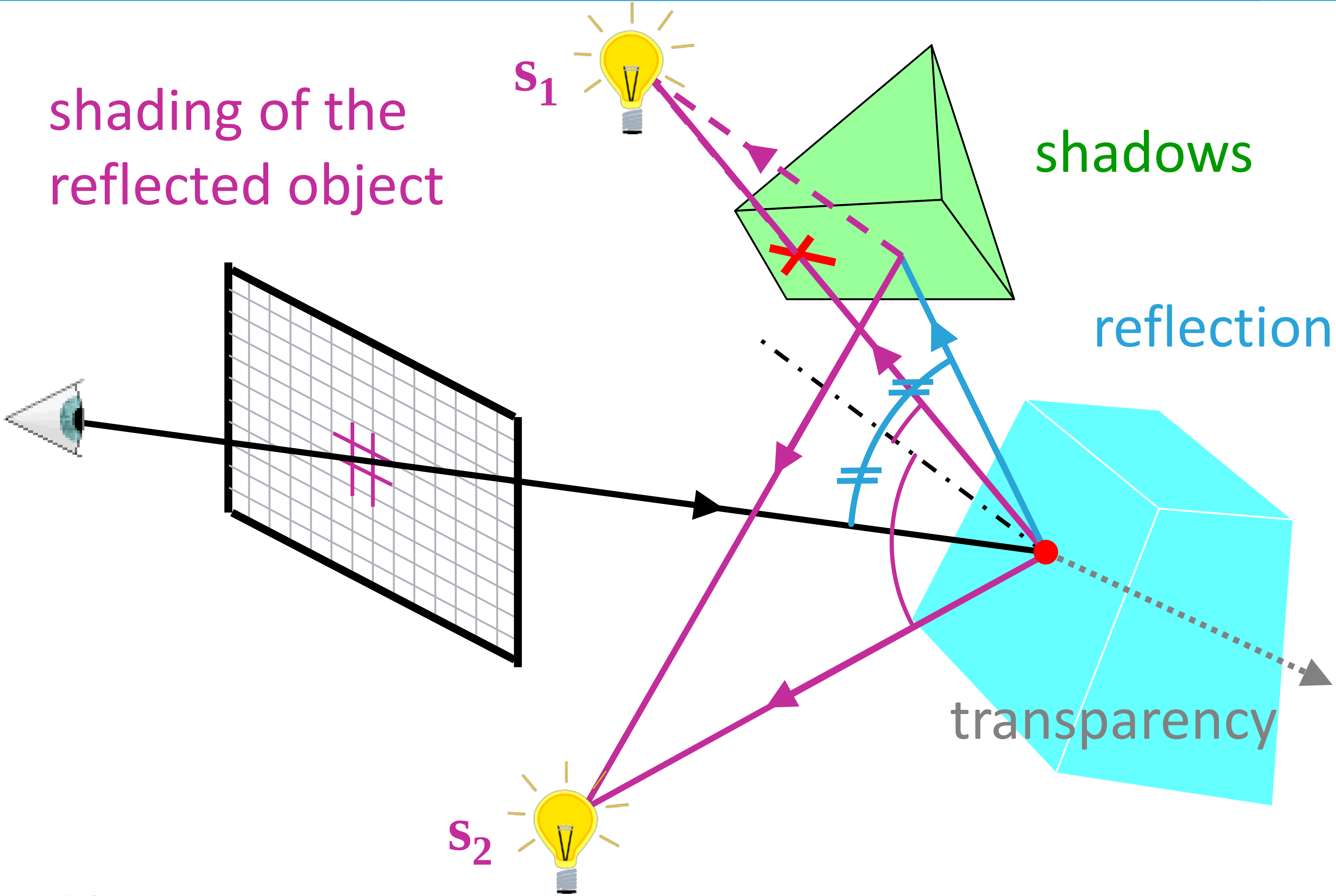
Ray Tracing in the Rendering Pipeline



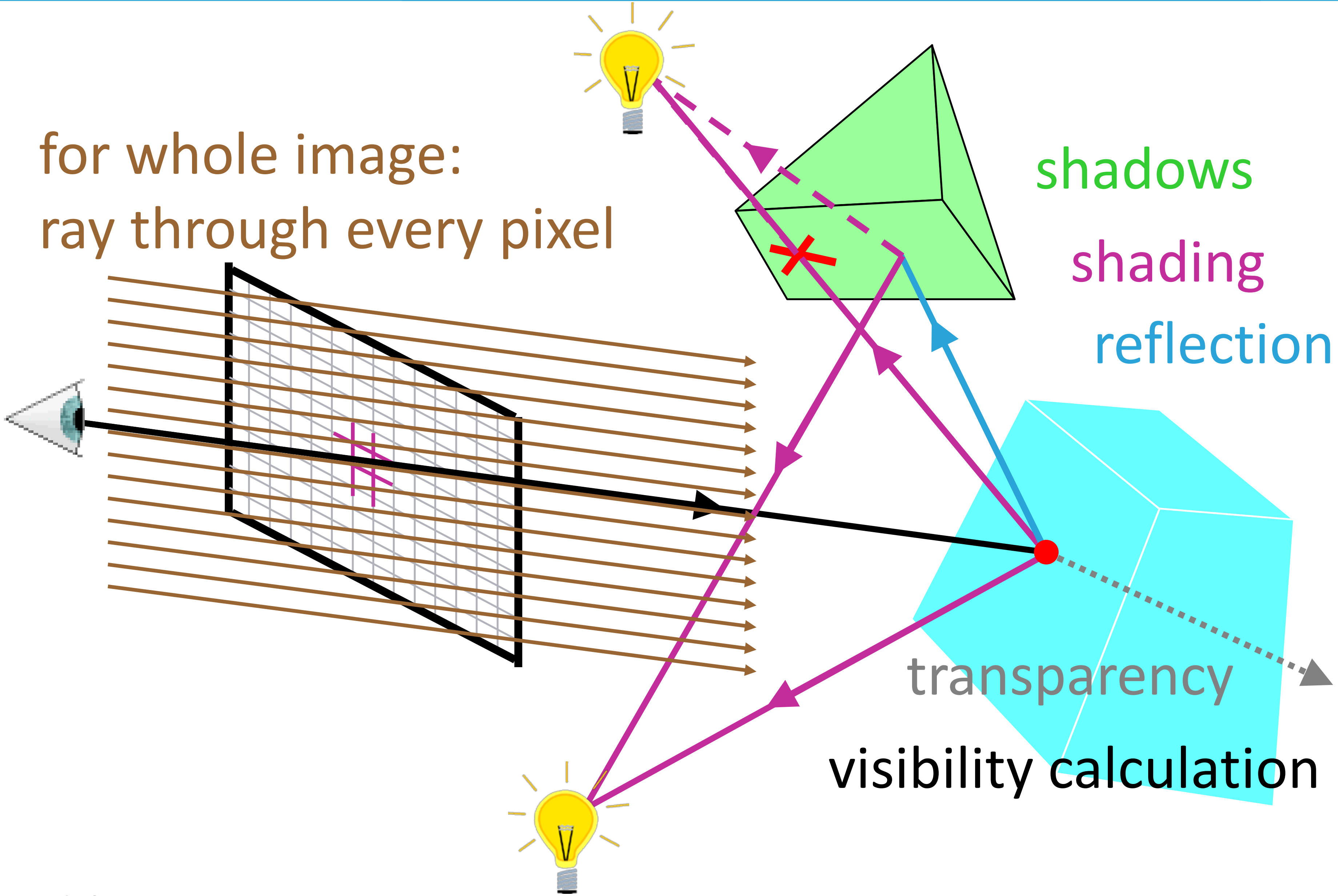
Ray Tracing Concepts



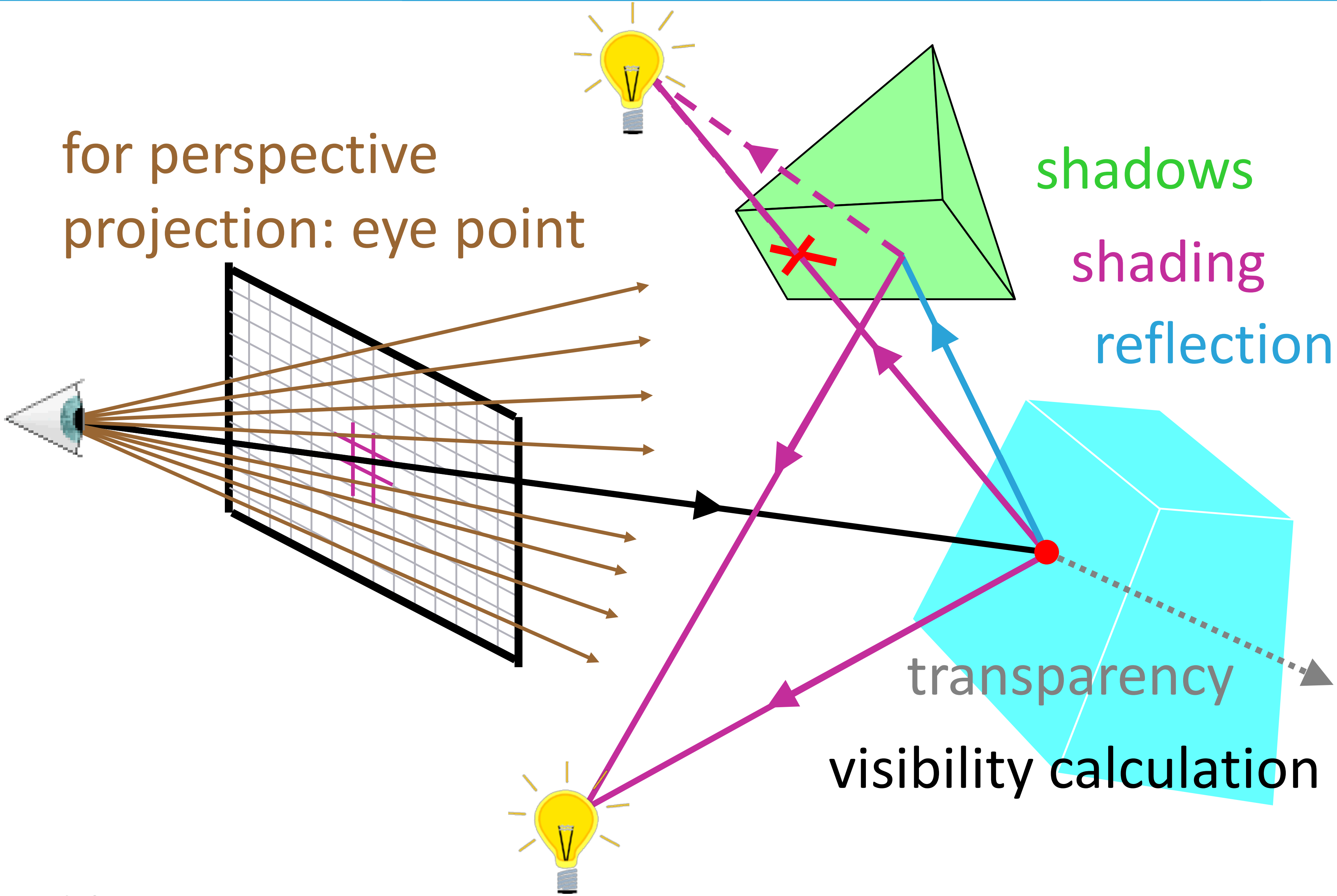
Ray Tracing Concepts



Ray Tracing Concepts

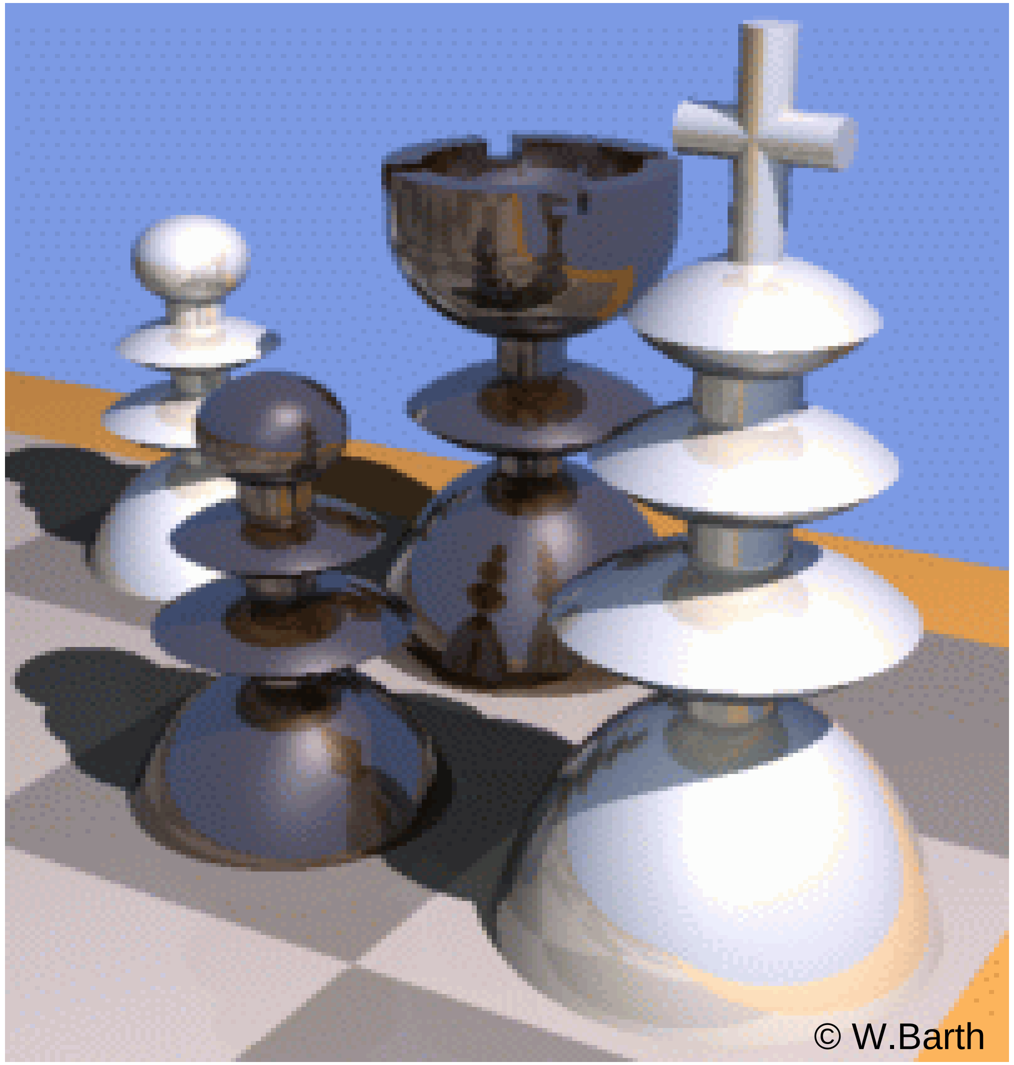


Ray Tracing Concepts



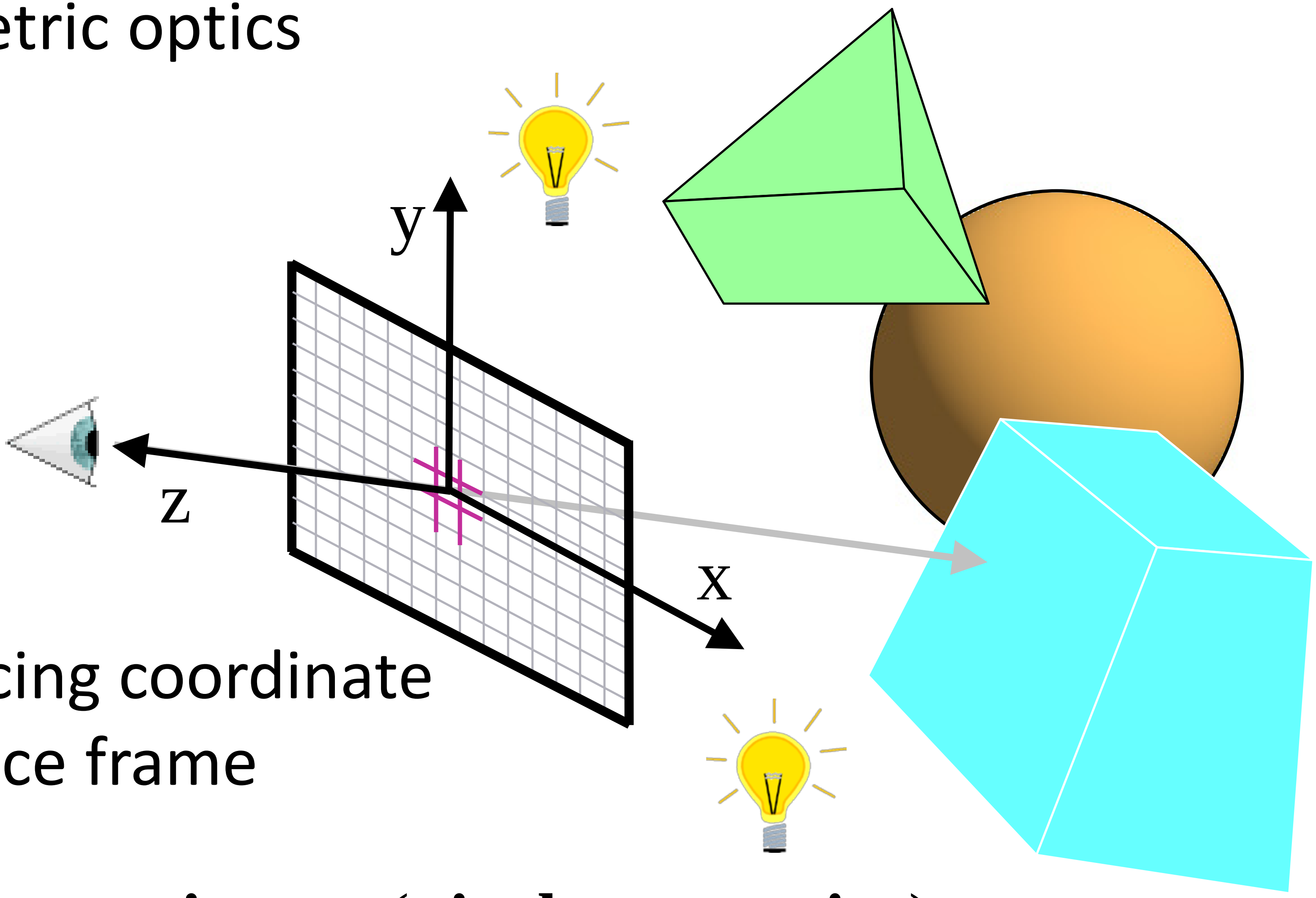
Ray Tracing Properties

- highly realistic images
- very time consuming
- multiple light sources
- visible-surface detection
- shadows
- reflections
- transparency



principles of geometric optics

projection
reference point



ray-tracing coordinate
reference frame

primary ray = **eye point** + t·(**pixel** – **eye point**)

Shading: Diffuse Shading

$$\boxed{\mathbf{I_d} = \text{xxx}}$$

I_d … illumination caused by diffuse shading

xxx … any shading model

(Phong, Blinn-Phong, Cook-Torrance,…)

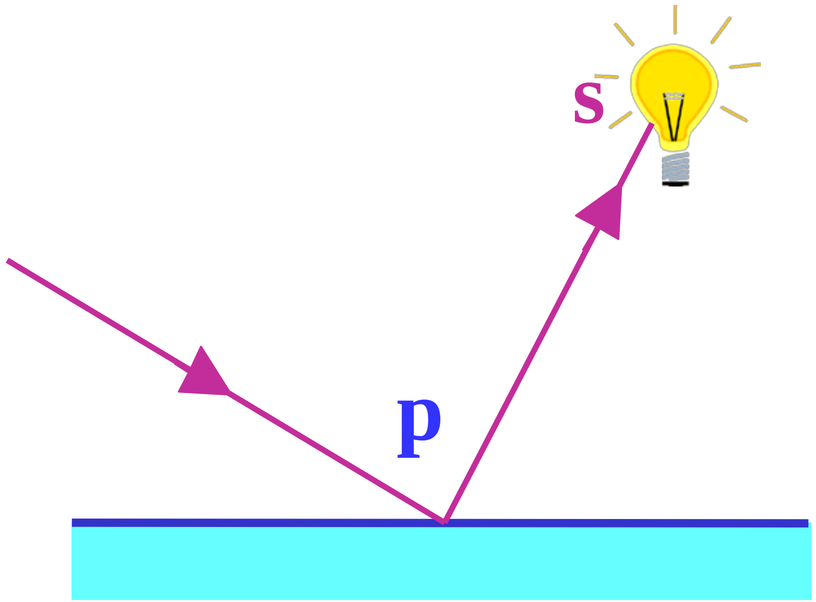
Ray Tracing: Shadows

ray = intersection point + t · vector to light source

$$\text{ray} = \mathbf{p} + t \cdot (\mathbf{s} - \mathbf{p})$$

$\mathbf{p}$ … intersection point

$\mathbf{s}$ … light source position

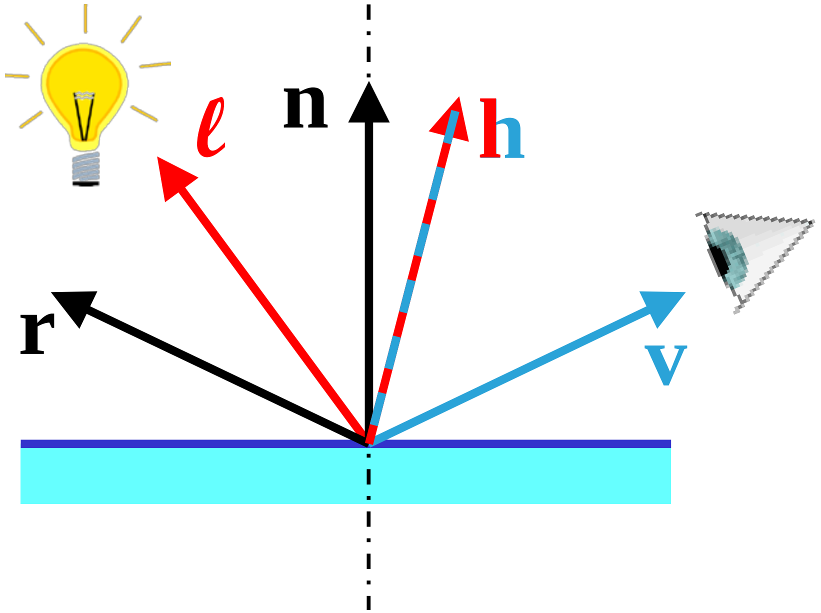


a light source influences the result only if there is no intersection with $0 < t < 1$

Ray Tracing: Shadows and Shading

- shadow ray along $\boldsymbol{\ell}$
- ambient light $\quad k_a I_a$
- diffuse reflection $\quad k_d(\mathbf{n} \cdot \boldsymbol{\ell})$
- specular reflection $\quad k_s(\mathbf{h} \cdot \mathbf{n})^p$

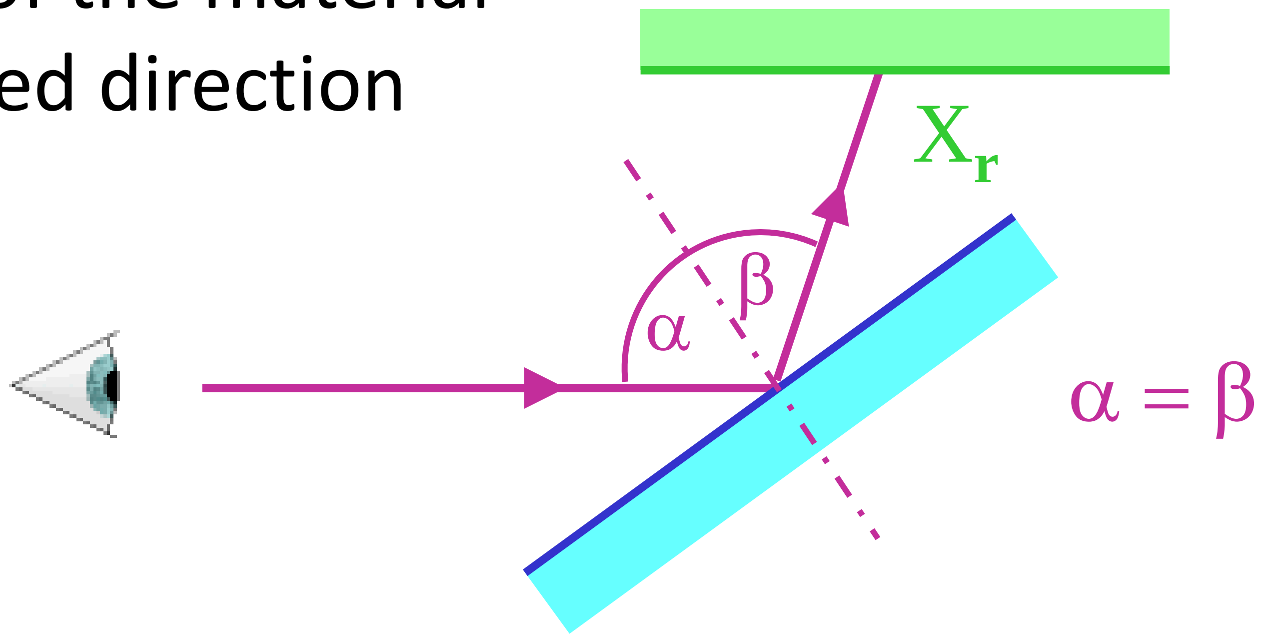
$$\boxed{I_d = k_a I_a + k_d(\mathbf{n} \cdot \boldsymbol{\ell}) + k_s(\mathbf{h} \cdot \mathbf{n})^p}$$



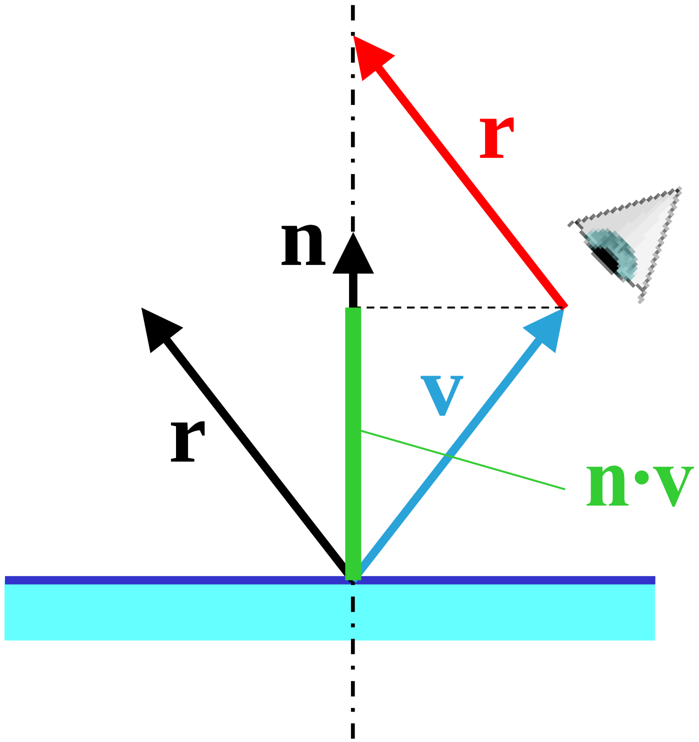
Ray Tracing: Reflection

$$\boxed{I_{\mathbf{r}} = k_{\mathbf{r}} \cdot X_{\mathbf{r}}}$$

I_r … illumination caused by reflection
k_r … reflection coefficient of the material
X_r … shading in the reflected direction



calculation of reflection ray

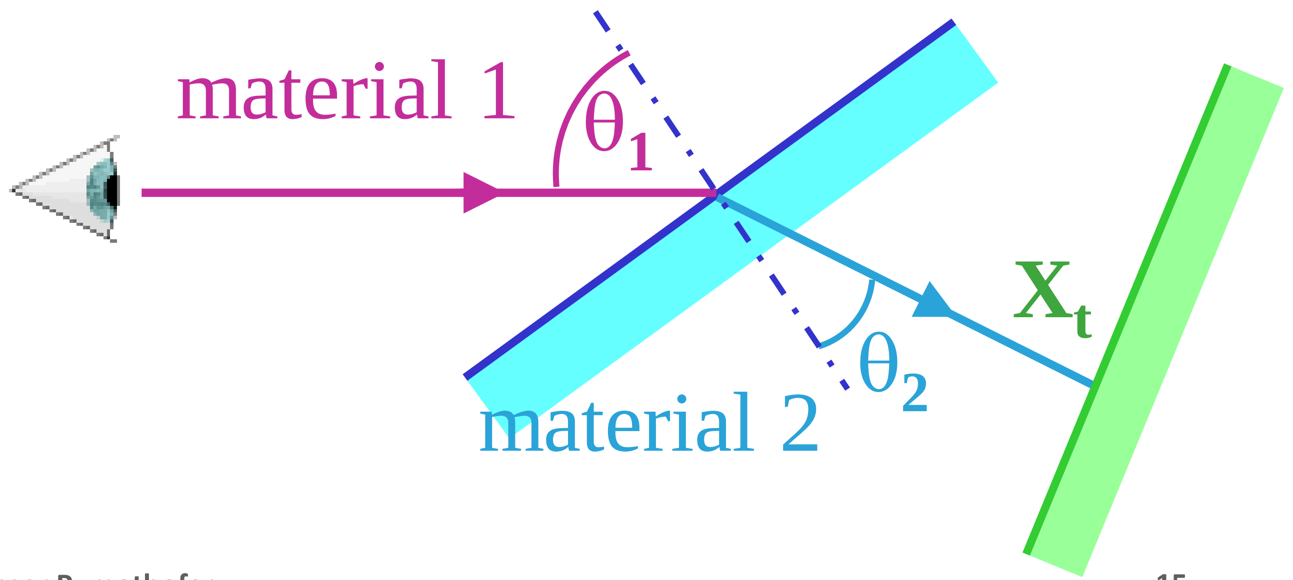


$$\mathbf{r} + \mathbf{v} = (2\mathbf{n}\cdot\mathbf{v})\mathbf{n}$$

$$\mathbf{r} = (2\mathbf{n}\cdot\mathbf{v})\mathbf{n} - \mathbf{v}$$

$$\boxed{I_t = k_t \cdot X_t}$$

I_t … illumination caused by transparency
k_t … transparency coefficient of the material
X_t … shading in the transparency direction

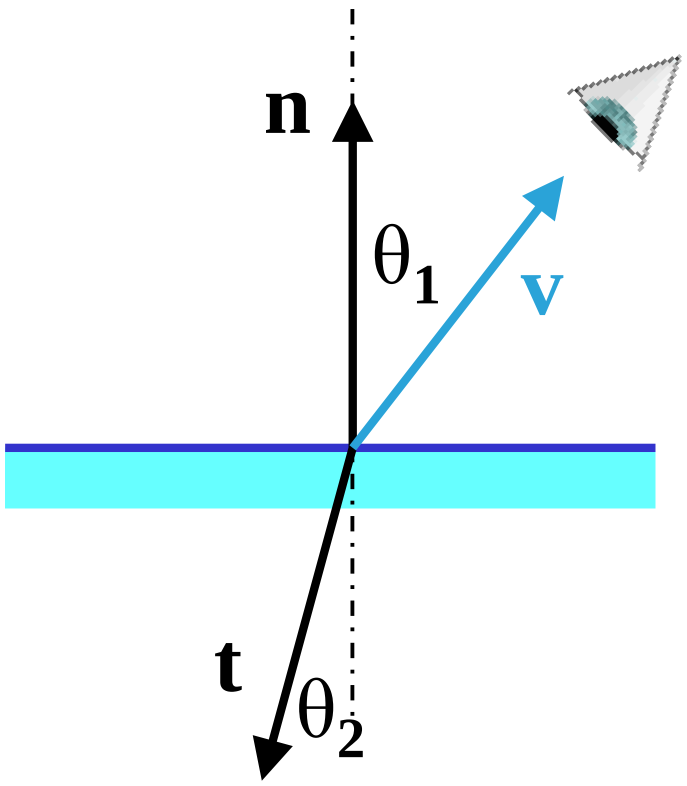


$$\sin\theta_1 : \sin\theta_2 = \eta_2 : \eta_1$$

calculation of transparency ray

$$\sin\theta_2 = \frac{\eta_1}{\eta_2}\sin\theta_1$$

$$\mathbf{t} = -\frac{\eta_1}{\eta_2}\mathbf{v} - \left(\cos\theta_2 - \frac{\eta_1}{\eta_2}\cos\theta_1\right)\mathbf{n}$$



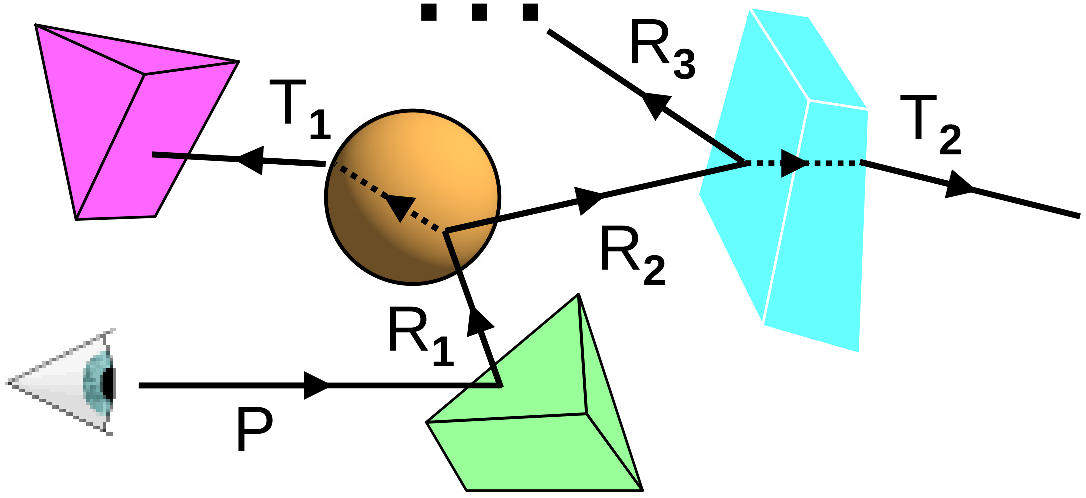
Ray Tracing: A Complete Shading Method

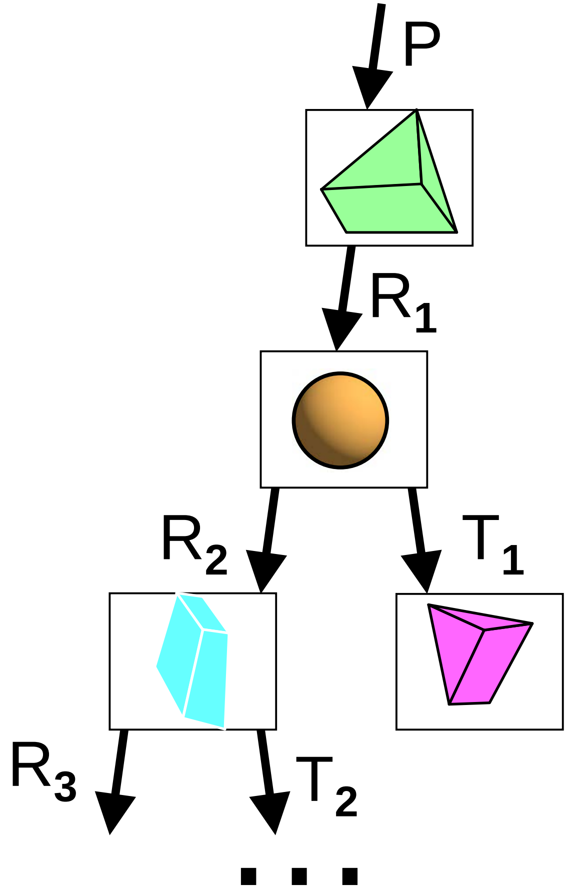
$$\boxed{I = I_{\mathbf{d}} + I_{\mathbf{r}} + I_{\mathbf{t}}}$$

additional requirement: $k_{\mathbf{d}} + k_{\mathbf{r}} + k_{\mathbf{t}} \leq 1$

Ray Tracing: Rays & Ray Tree

primary and secondary rays





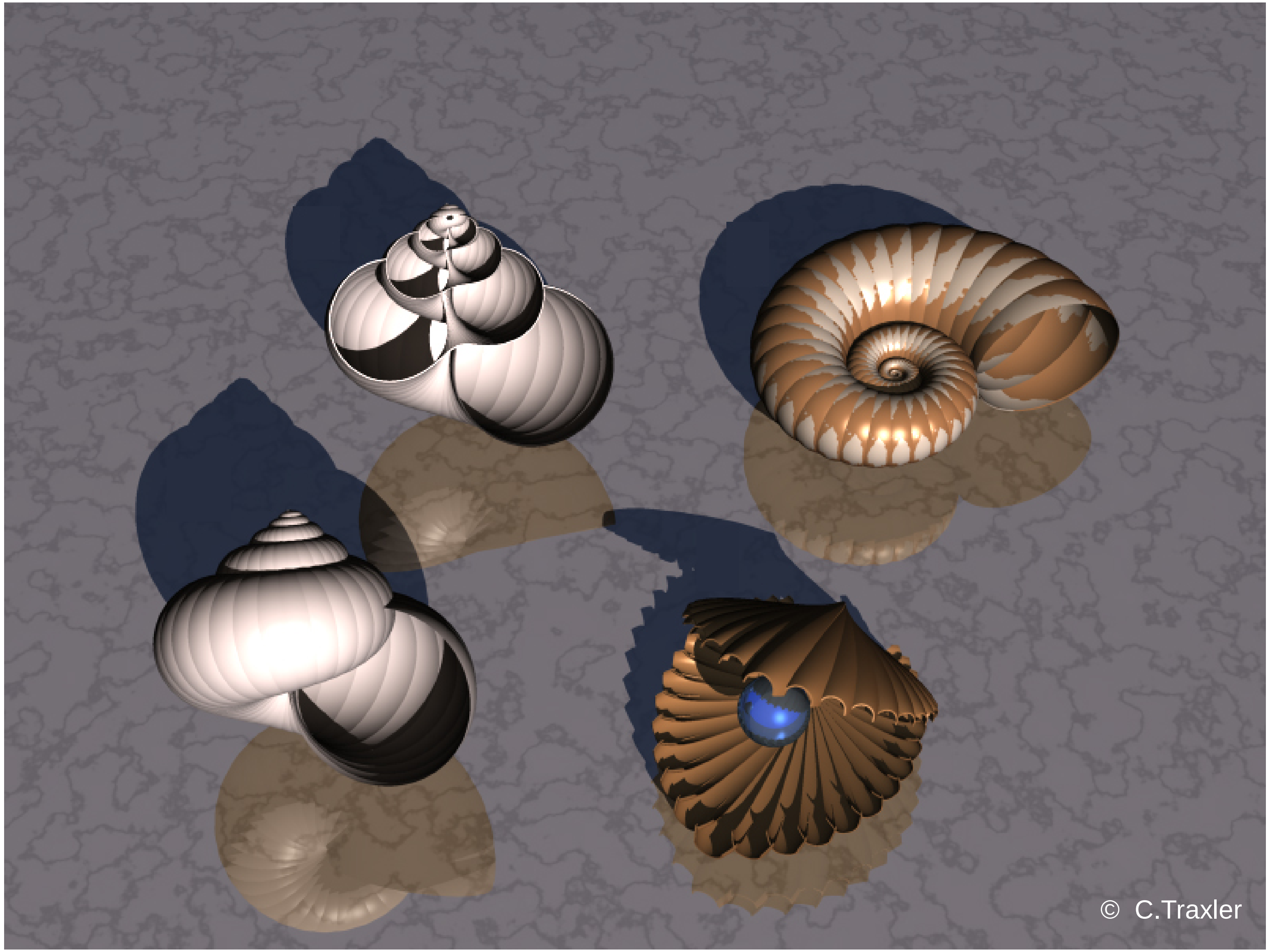
reflection and refraction ray paths for one pixel

corresponding binary ray tracing tree

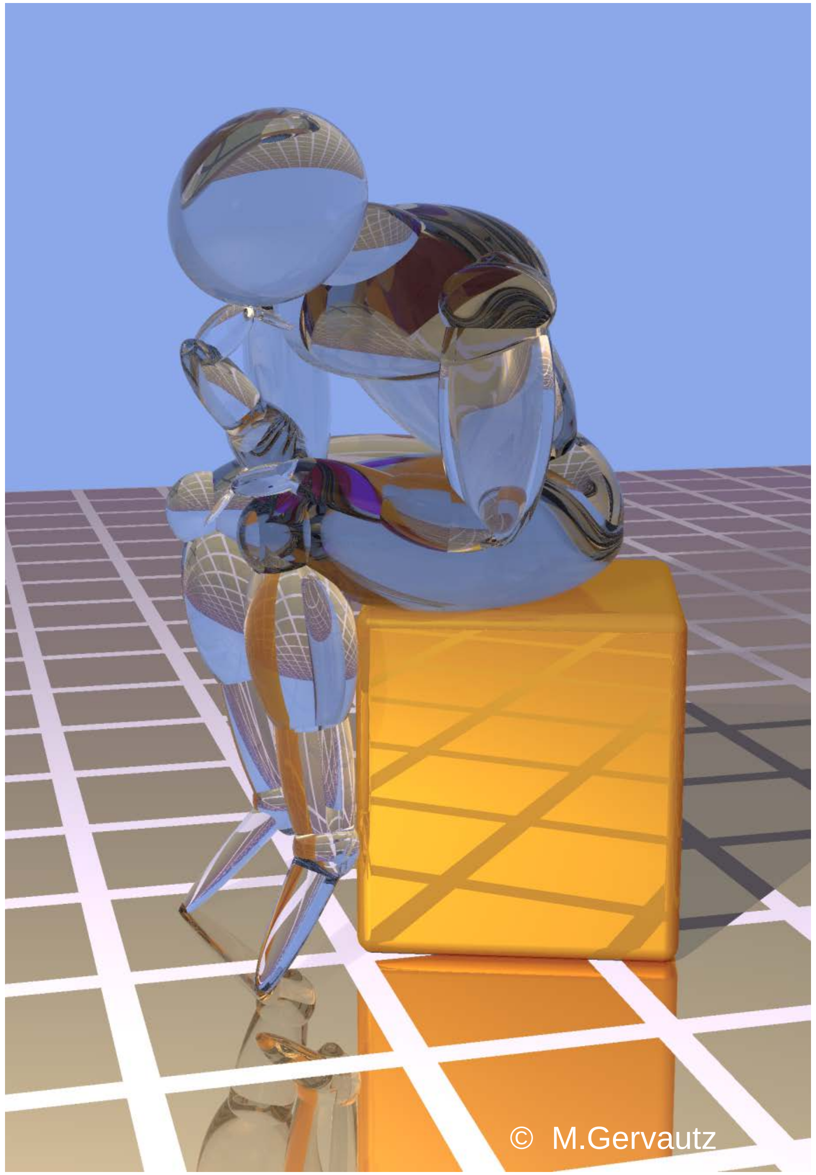
Ray Tracing: Basic Algorithm

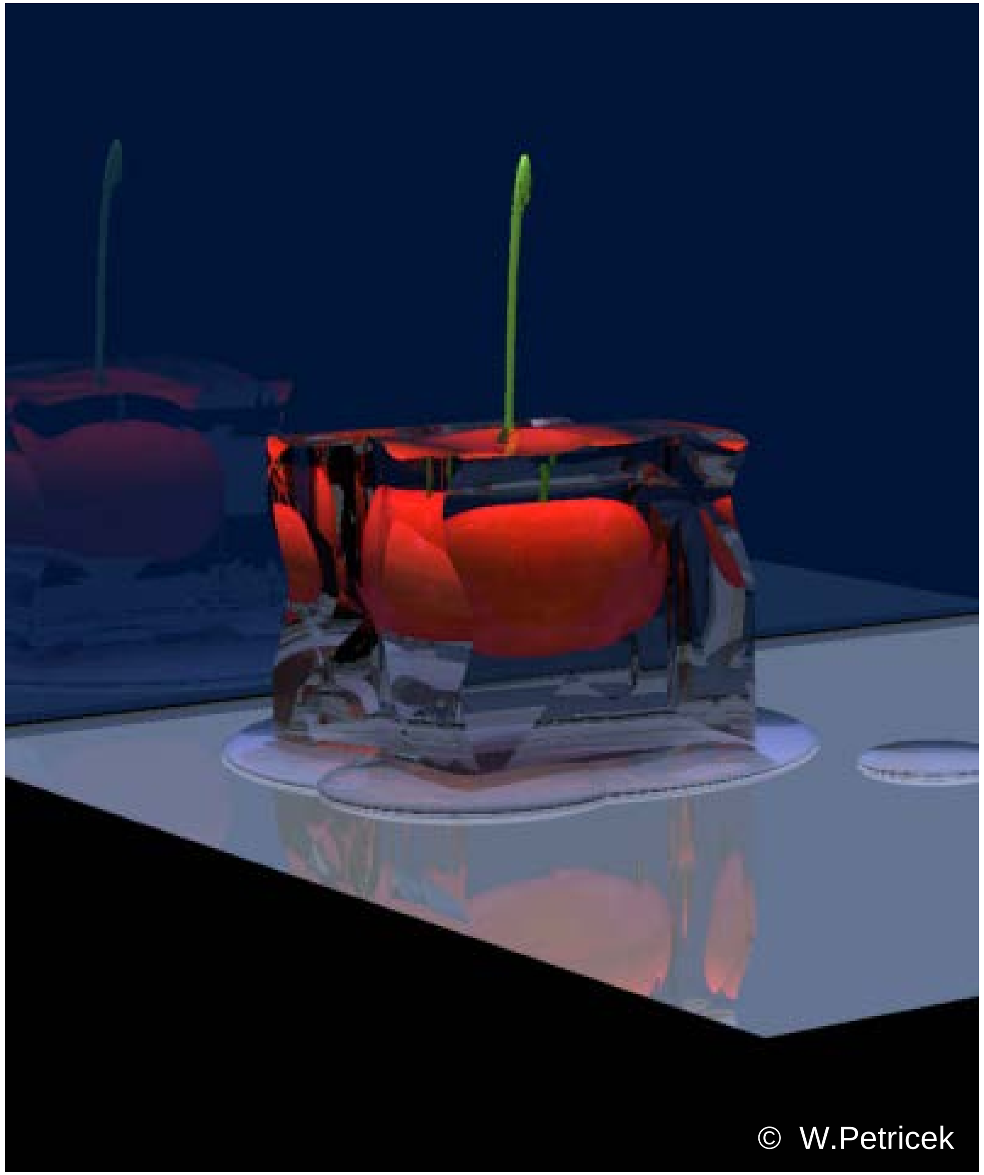
```
FOR all pixels p0 DO
1.trace primary ray from eye e to p0
  find closest intersection p
2.FOR all light sources s DO
    trace shadow feeler from p to s
    IF no intersection between p & s
    THEN shading += influence of s
3.IF surface of p is reflective
  THEN trace secondary ray;
    shading += influence of reflection
4.IF surface of p is transparent
  THEN trace secondary ray;
    shading += influence of transparency
```

Ray Tracing Examples

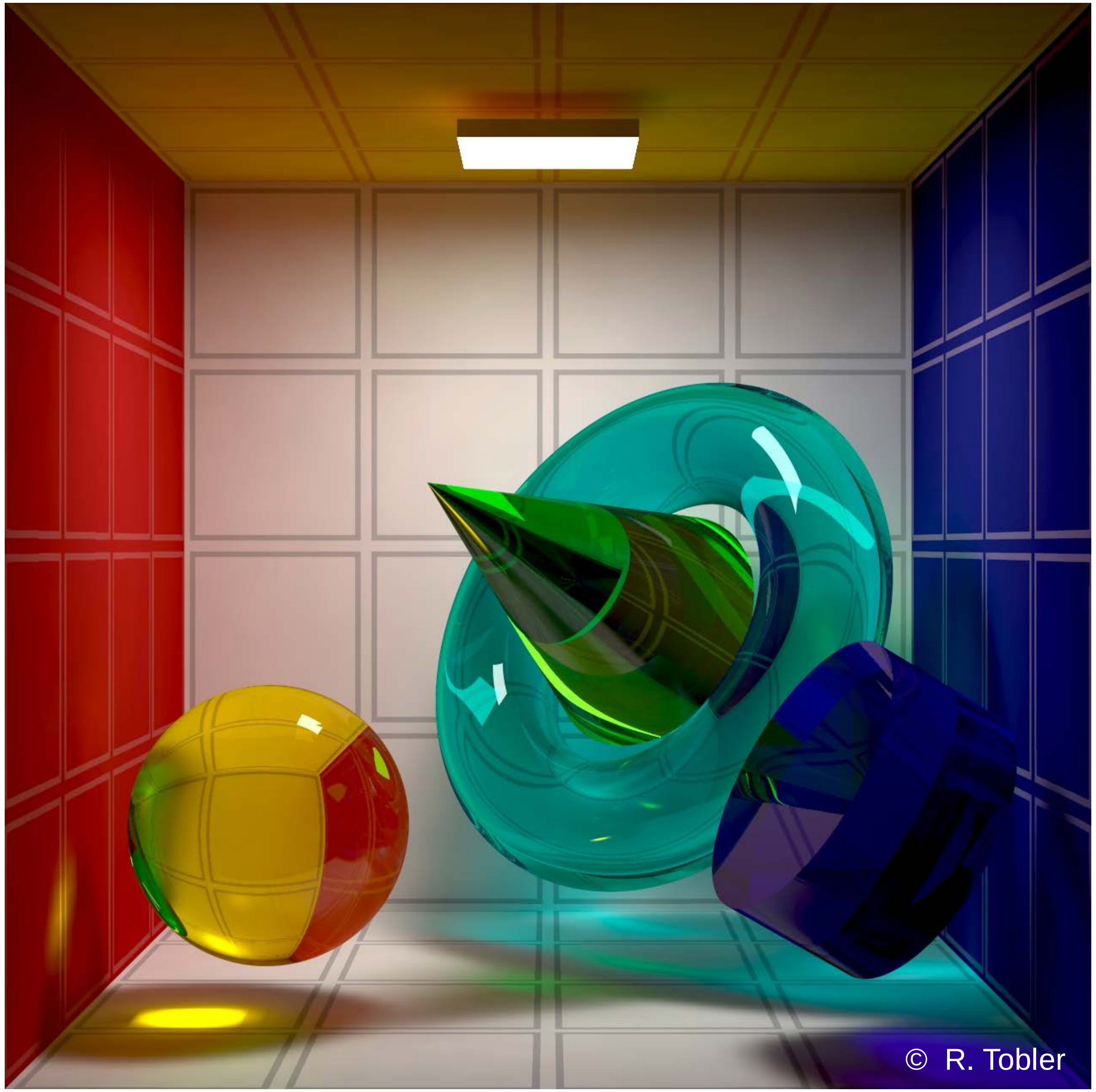


Ray Tracing Examples





True Global Illumination Example



Requirements for Object Data

(to use them for ray tracing)

1. intersection calculation ray ↔ object must be possible
2. surface normal calculation must be possible

→ simple for B-Rep

→ recursive evaluation for CSG

Ray-Surface Intersection

ray equation:

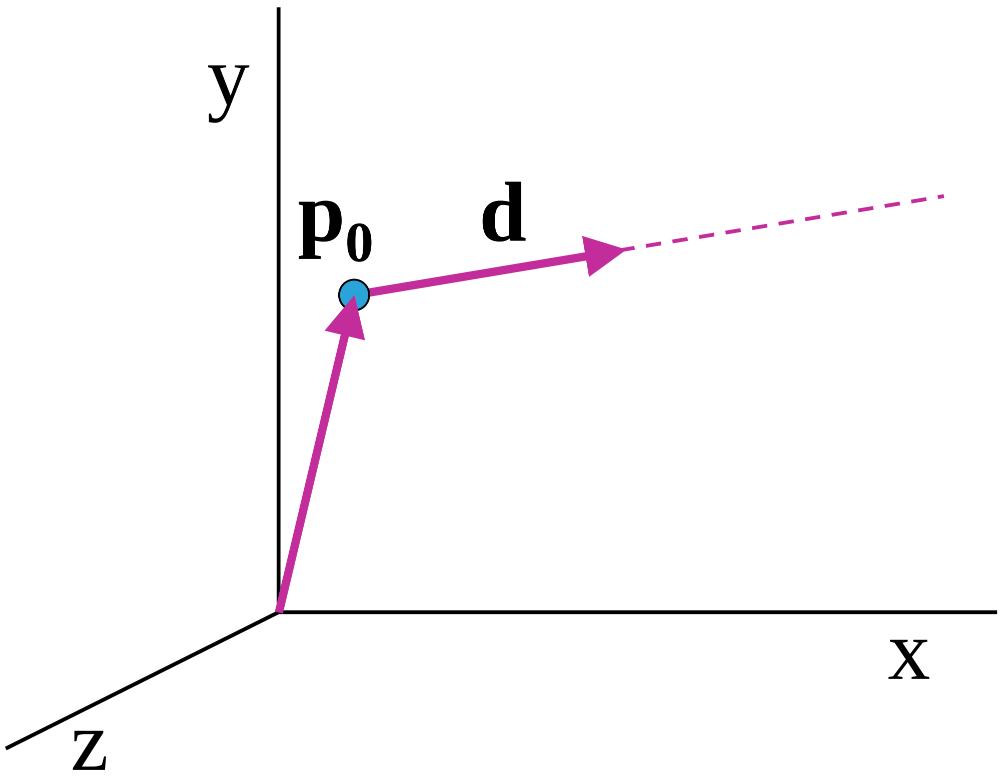
$$\mathbf{p}(t) = \mathbf{p}_0 + t \cdot \mathbf{d}$$

for primary rays:

$$\mathbf{d} = \frac{\mathbf{p}_0 - \mathbf{e}}{|\mathbf{p}_0 - \mathbf{e}|}$$

for secondary rays:

$$\mathbf{d} = \mathbf{r}$$
$$\mathbf{d} = \mathbf{t}$$



describing a ray with an initial-position vector $\mathbf{p}_0$ *and unit direction vector* $\mathbf{d}$

Ray-Sphere Intersection

parametric ray equation
inserted into sphere equation

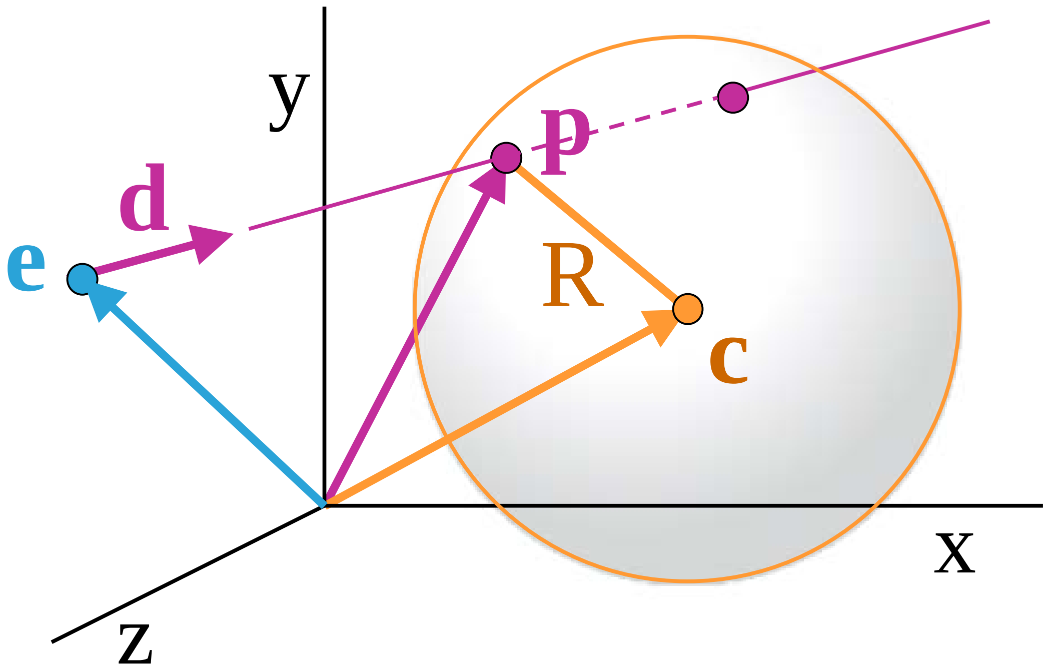
$$|\mathbf{p} - \mathbf{c}|^2 - R^2 = 0$$

$$|(\mathbf{e} + t\mathbf{d}) - \mathbf{c}|^2 - R^2 = 0$$

$$\Delta\mathbf{p} = \mathbf{c} - \mathbf{e}$$

$$t^2 - 2(\mathbf{d} \cdot \Delta\mathbf{p})t + \left(|\Delta\mathbf{p}|^2 - R^2\right) = 0 \qquad (\mathbf{d}^2 = 1)$$

$$t = \mathbf{d} \cdot \Delta\mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \Delta\mathbf{p})^2 - |\Delta\mathbf{p}|^2 + R^2}$$

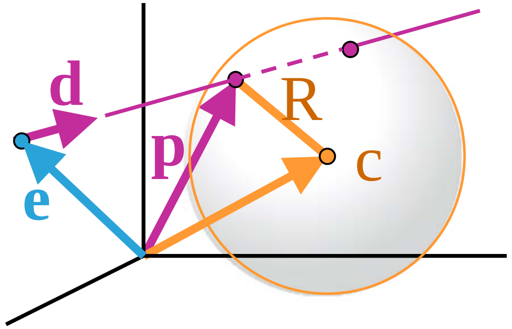


Ray-Sphere Intersection

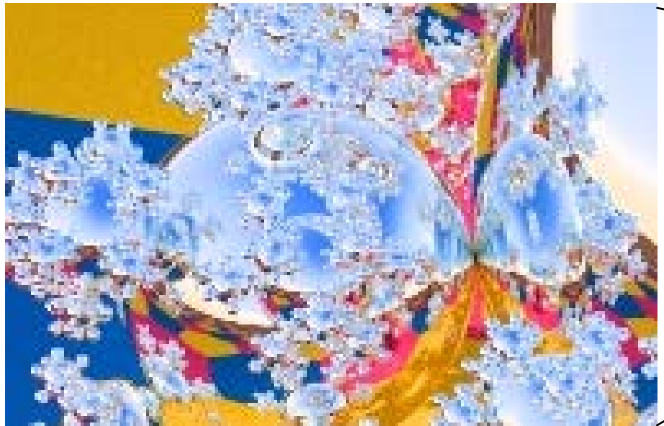
- discriminant negative $\Rightarrow$ no intersections

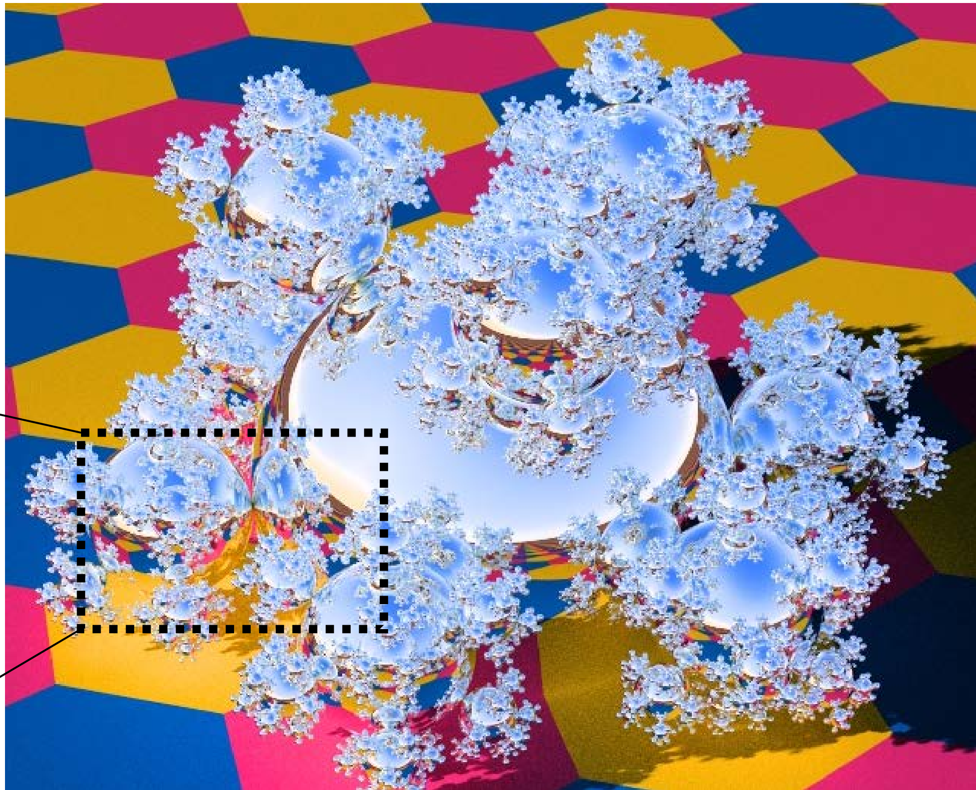
$$t = \mathbf{d}\cdot\Delta\mathbf{p} \pm \sqrt{(\mathbf{d}\cdot\Delta\mathbf{p})^2 - |\Delta\mathbf{p}|^2 + R^2}$$

→ roundoff errors
when $R^2 << |\Delta\mathbf{p}|^2$



"sphereflake"





Ray-Sphere Intersection

- discriminant negative ⇒ no intersections

$$t = \mathbf{d}\cdot\Delta\mathbf{p} \pm \sqrt{(\mathbf{d}\cdot\Delta\mathbf{p})^2 - |\Delta\mathbf{p}|^2 + R^2}$$

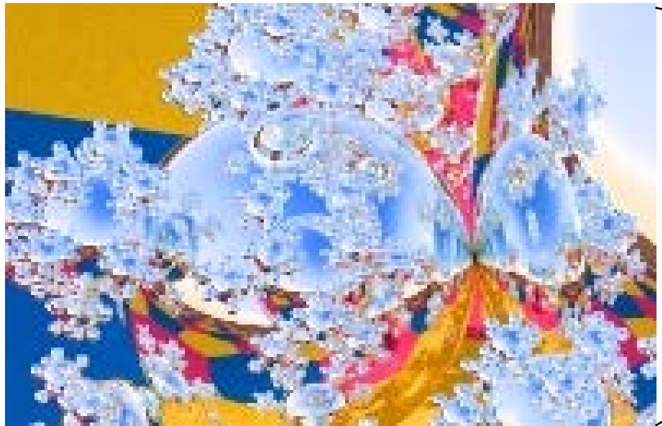
$$|\Delta p - (d\cdot\Delta p)d|^2 =$$
$$= |\Delta p|^2 - 2d^2|\Delta p|^2 + (d\cdot\Delta p)^2 d^2$$

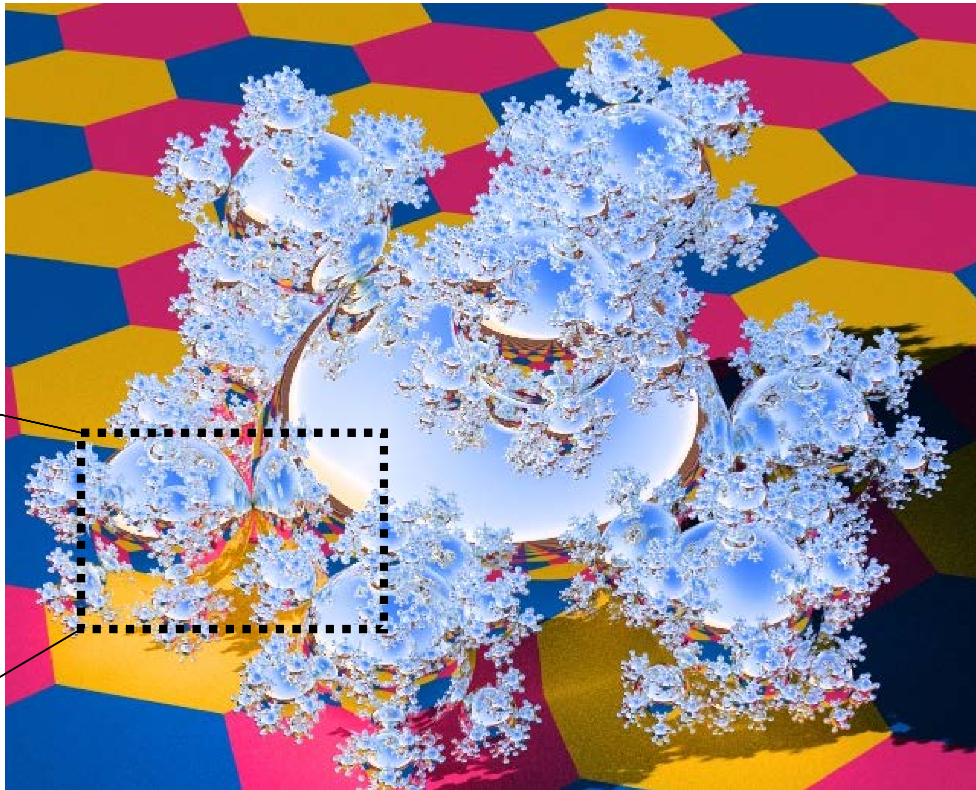
➡ $$t = \mathbf{d}\cdot\Delta\mathbf{p} \pm \sqrt{R^2 - |\Delta\mathbf{p} - (\mathbf{d}\cdot\Delta\mathbf{p})\mathbf{d}|^2}$$

because $\mathbf{d}^2=1$

(to avoid roundoff errors when $R^2 << |\Delta\mathbf{p}|^2$)

"sphereflake"





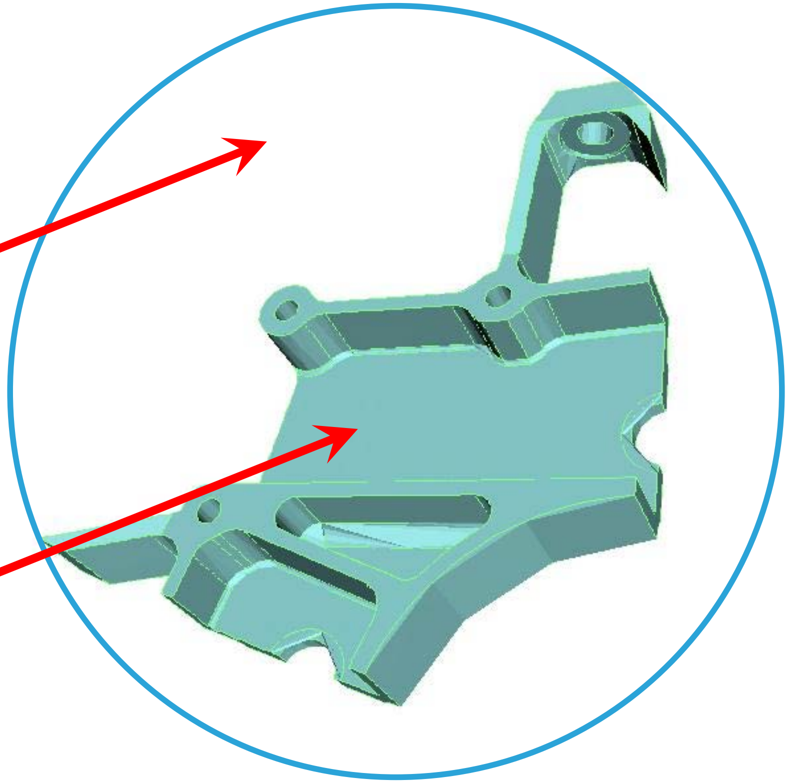
Ray-Polyhedron Intersection

use ***bounding sphere*** to eliminate easy cases

ray does not hit bounding sphere
→ no intersection with object

further investigation necessary:

- ray hits bounding sphere but no intersection with object
- ray hits bounding sphere and intersection with object



Ray-Polyhedron Intersection

- use bounding sphere to eliminate easy cases
- locate front faces $\quad \mathbf{d} \cdot \mathbf{n} < 0$
- solve plane equation

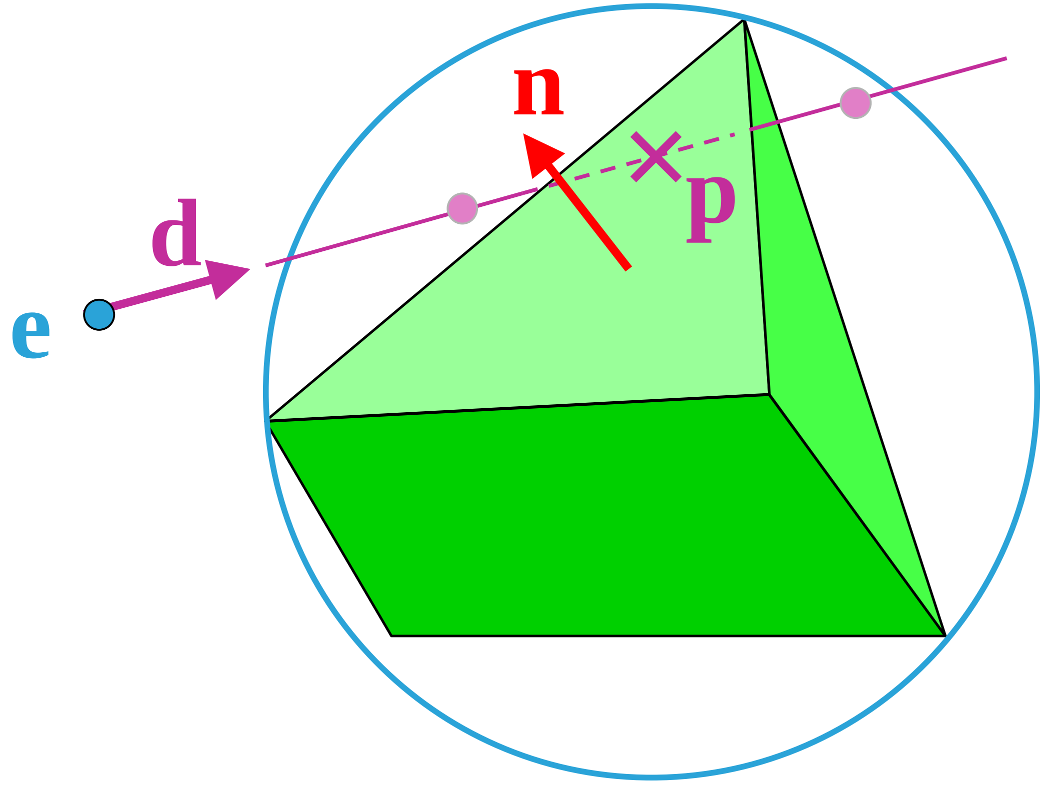
$$Ax + By + Cz + D = 0$$

$$\mathbf{n} = (A, B, C)$$

$$\mathbf{n} \cdot \mathbf{p} = -D$$

$$\mathbf{n} \cdot (\mathbf{e} + t\mathbf{d}) = -D$$

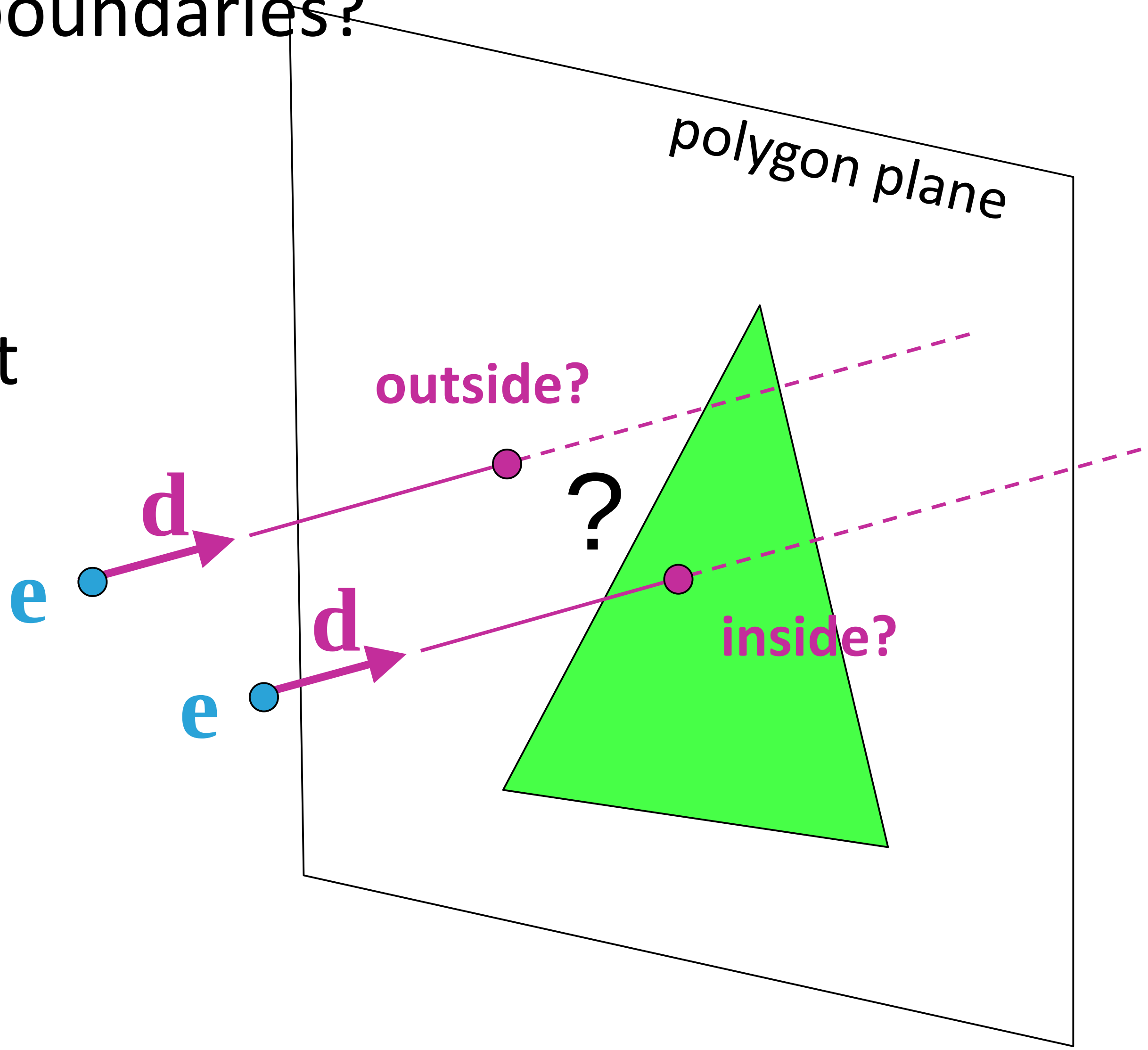
$$t = -\frac{D + \mathbf{n} \cdot \mathbf{e}}{\mathbf{n} \cdot \mathbf{d}}$$



intersection point inside polygon boundaries?

perform inside-outside test

→ smallest t to inside point is first intersection point of polyhedron

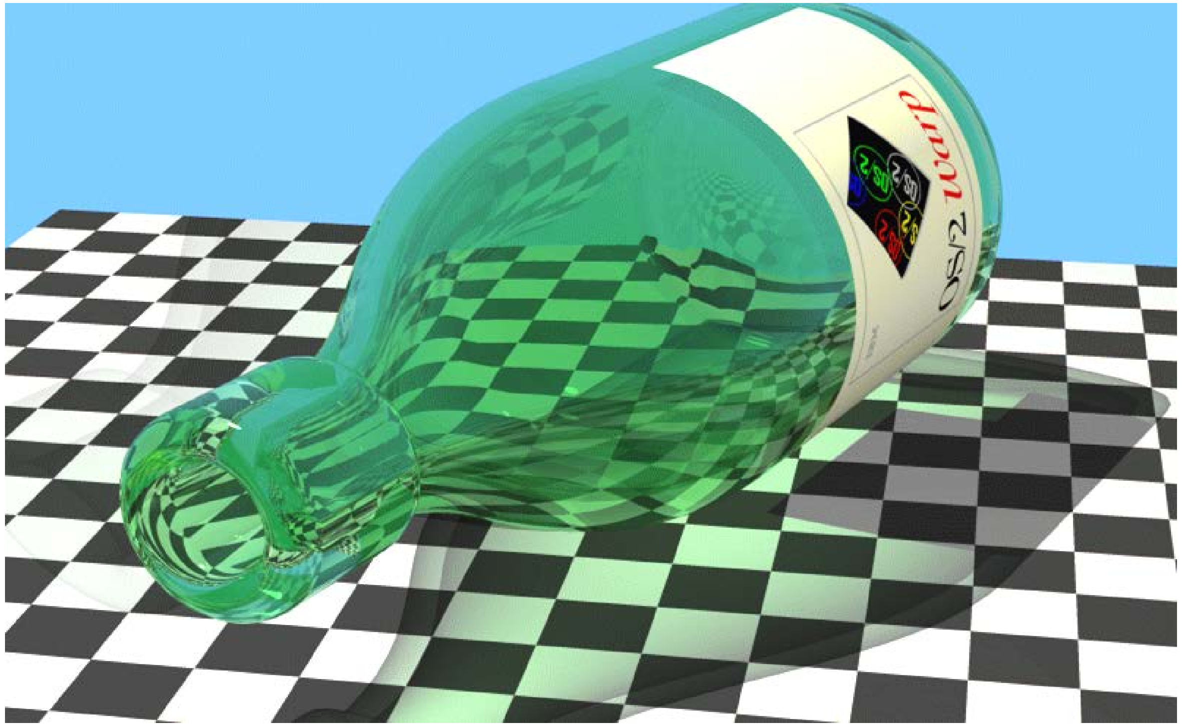


Ray-Surface Intersection

quadric, spline surfaces:

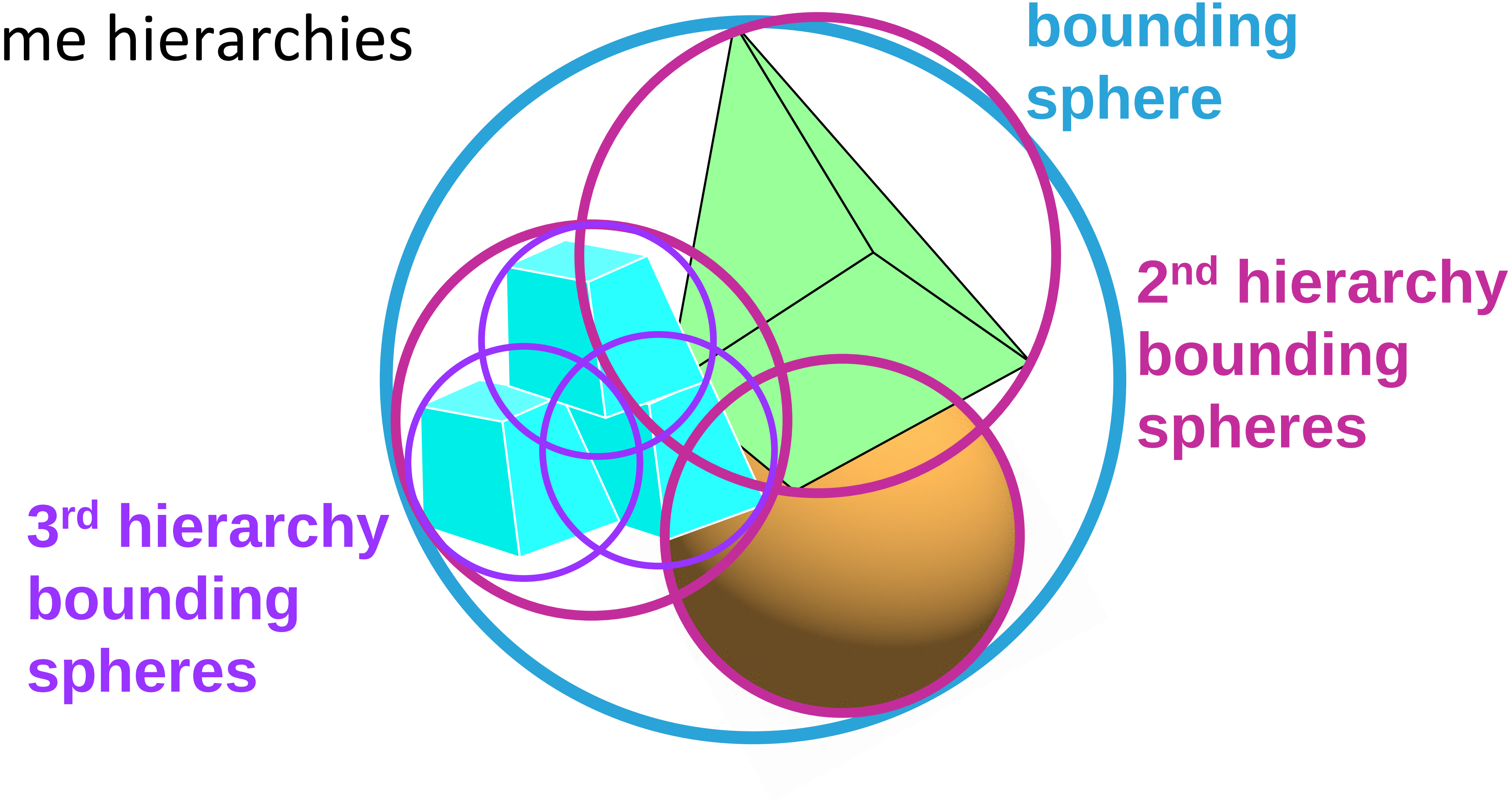
- parametric ray equation inserted into surface definition
- methods like numerical root-finding, incremental calculations

ray-traced scene with NURBS surfaces and multiple reflection / refraction



bounding volumes and

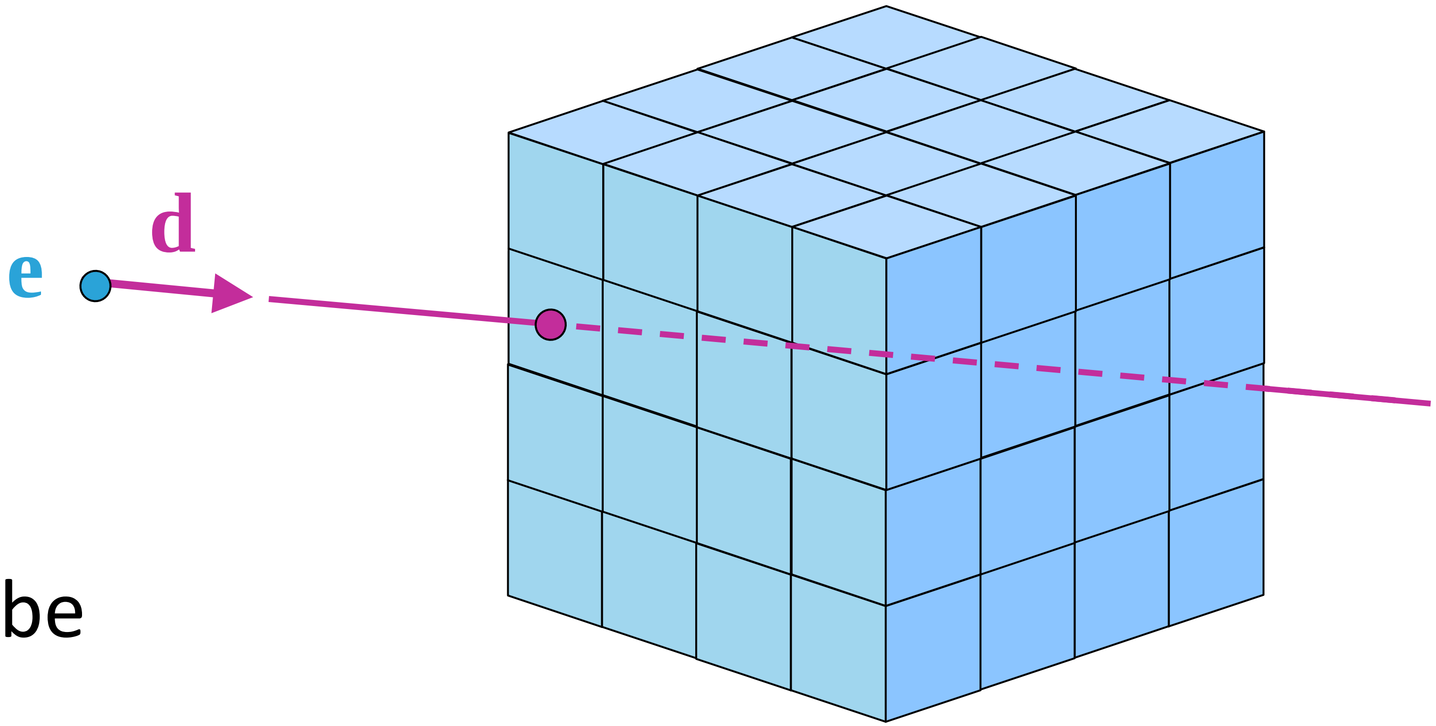
bounding volume hierarchies



Reducing Object-Intersection Calculations

space-subdivision methods

- regular grid
- octree

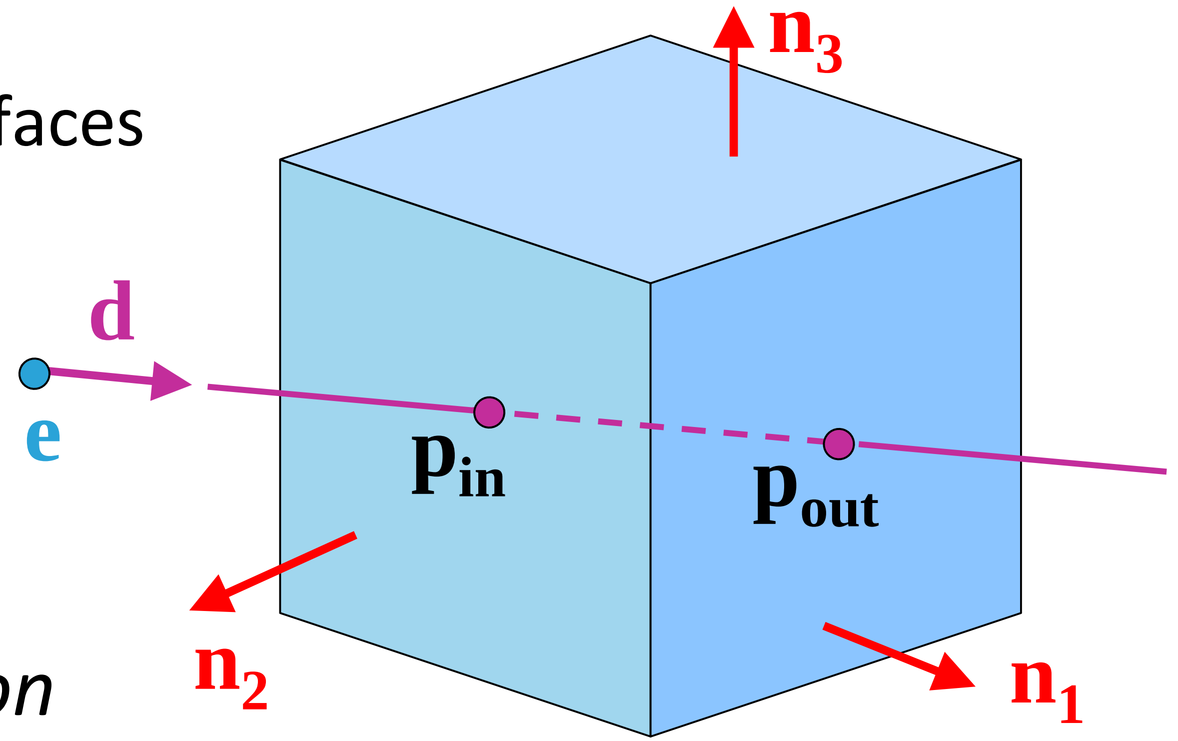


preprocess:
find object data in each cube

Reducing Object-Intersection Calculations

space-subdivision methods

- incremental grid traversal
  - 3D Bresenham
  - processing of potential exit faces

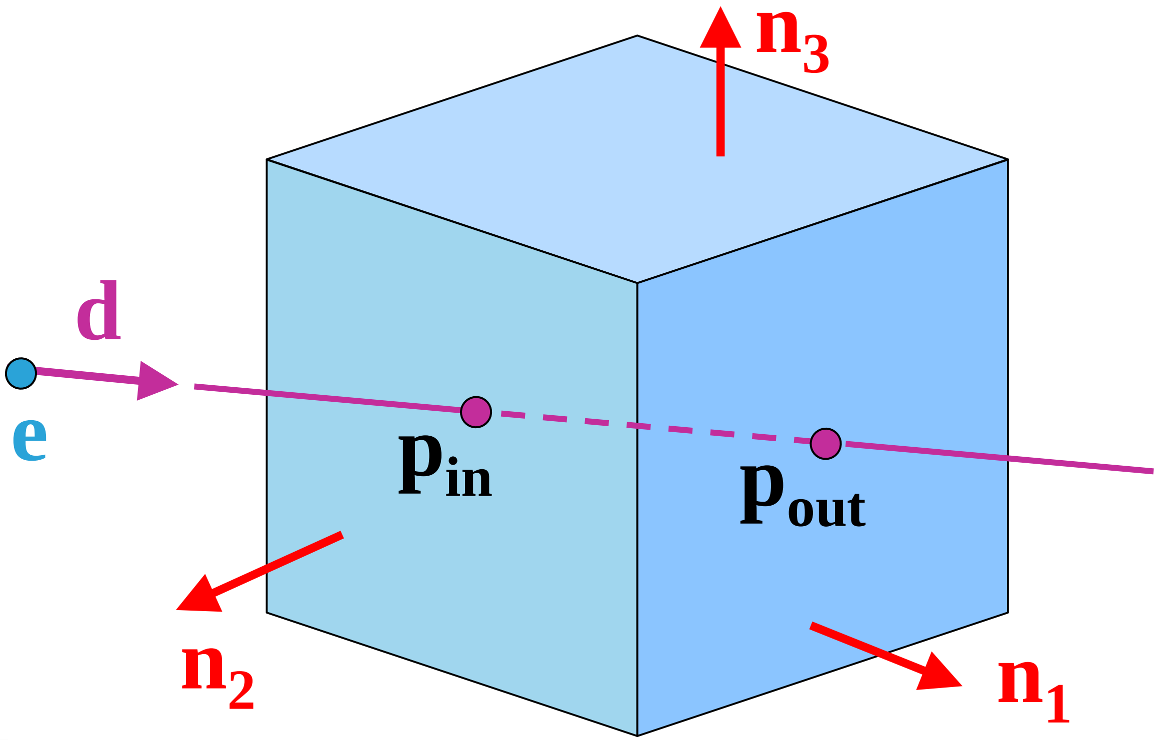


ray traversal through a subregion of a cube enclosing a scene

Incremental Grid Traversal

- ray direction $\mathbf{d}$ / ray entry position $\mathbf{p}_{in}$
- potential exit faces $\mathbf{d} \cdot \mathbf{n}_k > 0$
- normal vectors

$$\mathbf{n}_k = \begin{cases} (\pm 1, 0, 0) \\ (0, \pm 1, 0) \\ (0, 0, \pm 1) \end{cases}$$



- check signs of components of $\mathbf{d}$

calculation of exit positions, select smallest t_k

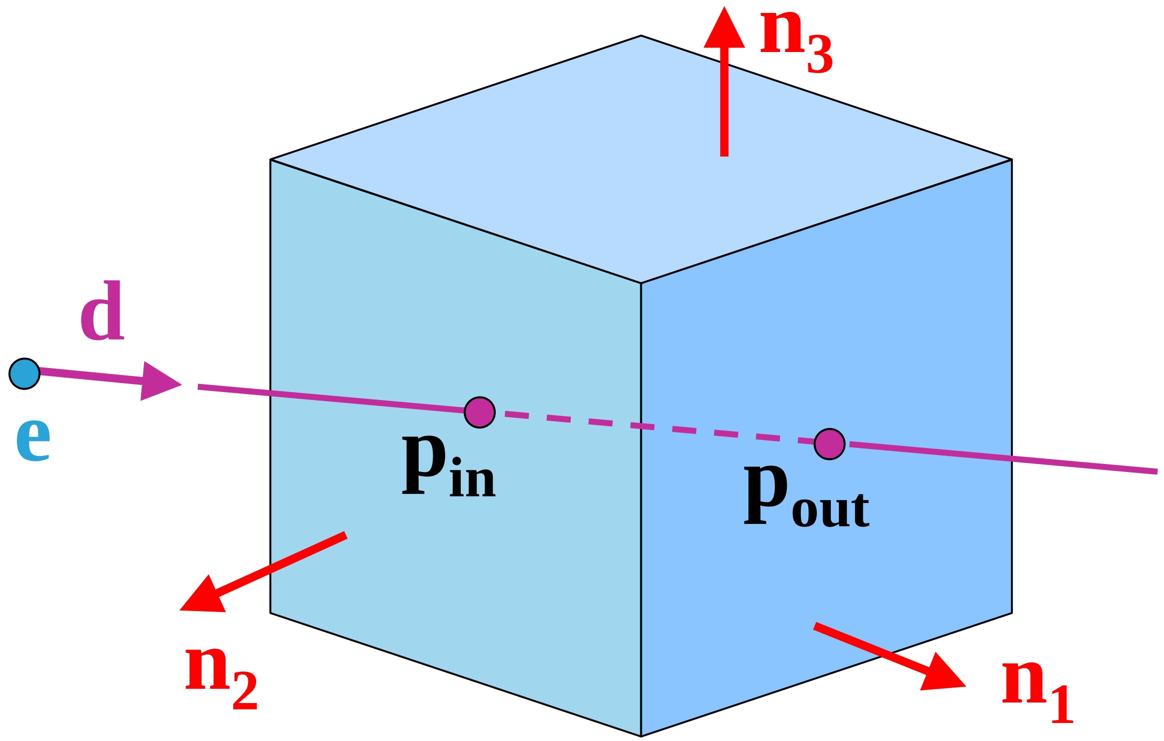
$$\mathbf{p}_{out,k} = \mathbf{p}_{in} + t_k\mathbf{d}$$

$$\mathbf{n}_k \cdot \mathbf{p}_{out,k} = -D_k$$

$$t_k = \frac{-D_k - \mathbf{n}_k \cdot \mathbf{p}_{in}}{\mathbf{n}_k \cdot \mathbf{d}}$$

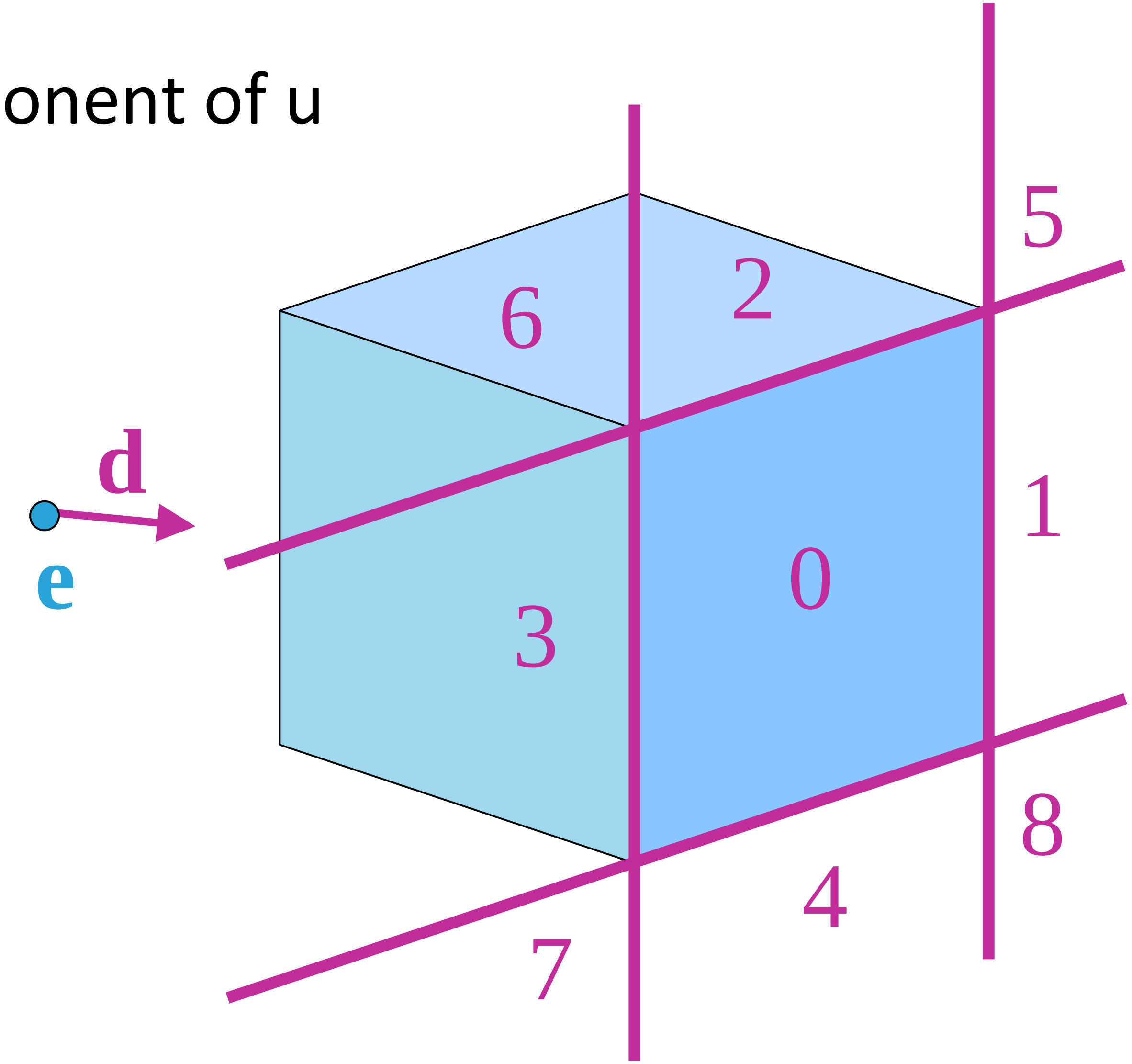
example: $\mathbf{n_k} = (1,0,0)$

$$x_\mathbf{k} = -D_\mathbf{k} \Rightarrow t_k = \frac{x_k - x_0}{x_d}$$

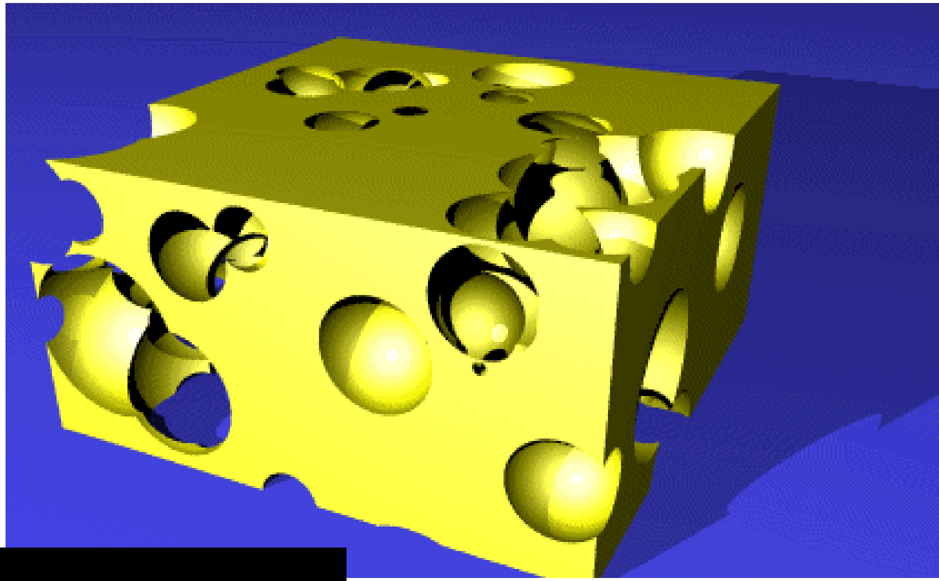


variation: trial exit plane

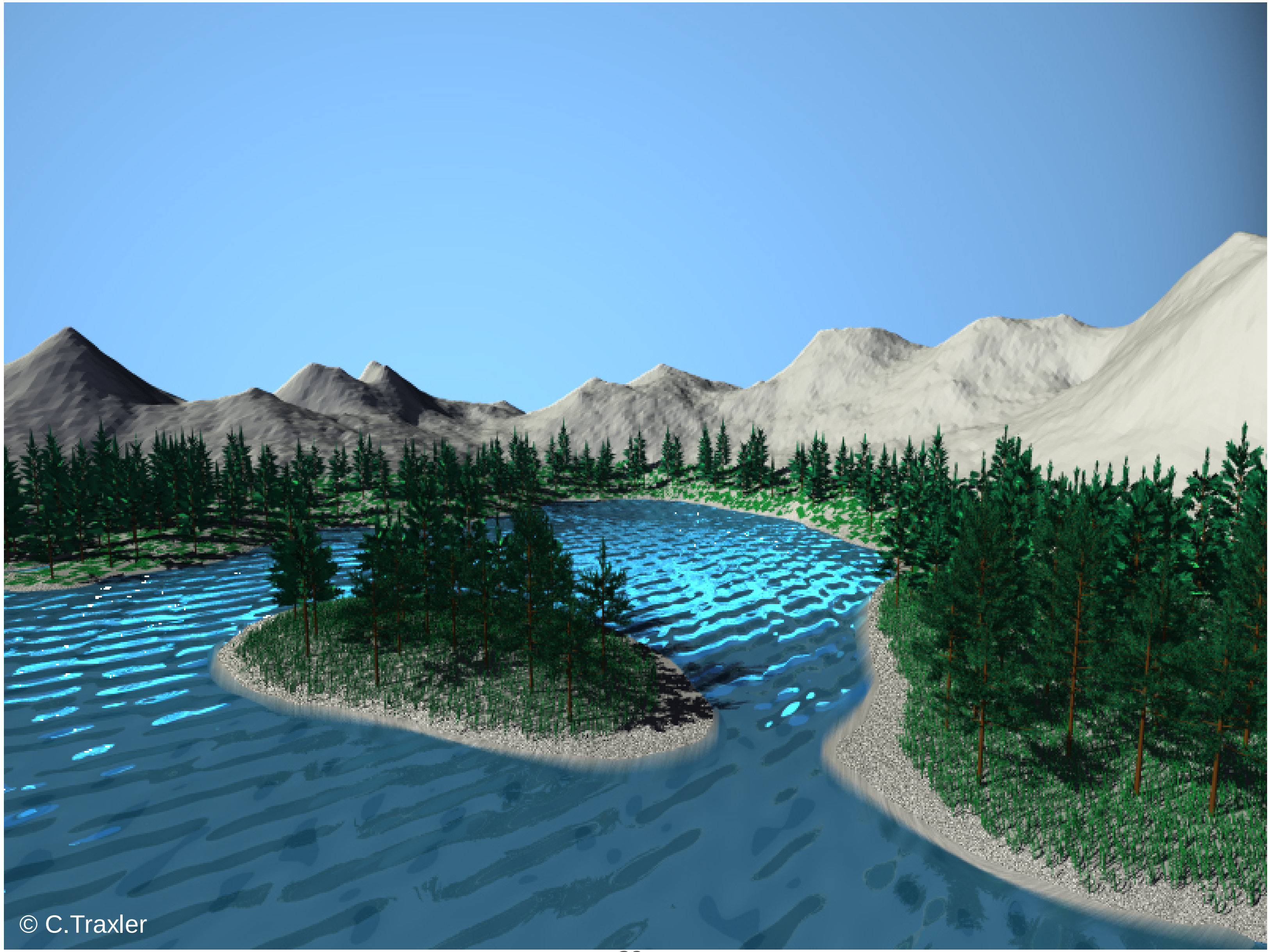
- perpendicular to largest component of u
- exit point in 0 $\Rightarrow$ done
- {1, 2, 3, 4} $\Rightarrow$ side clear
- {5, 6, 7, 8} $\Rightarrow$ extra calc.



Ray Tracing Examples



Ray Tracing Example



1 Billion Ray Traced Triangles